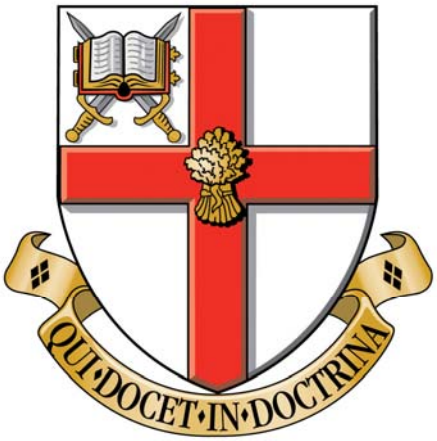

University of Chester

Effective Prediction of Intercensal Population Levels

Stewart John Norton

October 1999

Abstract

This dissertation is concerned with the prediction of population levels for the years following a census until the next census is counted. We review standard interpolation and extrapolation techniques, population models and neural networks. The population levels are required by government for allocating money to local authorities for spending on local services. The project was initiated by Chester City Council who consider that an underestimation of the Chester levels is causing a shortfall in the city's allocation of money.

Four credit dissertation submitted to Chester College for the

Degree of Master of Science (Mathematics) in part fulfilment of the

Modular Programme in Mathematics

I would like to thank the Mathematics Department of Chester College for offering a wide range of interesting modules on the Post Graduate Programme.

I would also like to thank:-

Dr. Neville Ford for his support throughout the course and his advice and encouragement in the preparation of this dissertation.

Chester City Council staff for supplying details of the Standard Spending Assessment and other relevant government information.

Kathy Sayer, University of Essex, and Roma Chappell, Office for National Statistics, for supplying census information and population data.

The staff of Bridgwater Public Library, Somerset for allowing me open access to their archives for research purposes.

The work of this dissertation is original and has not been submitted previously in support of any qualification or course.

Signed:

Date: 99

Contents

Chapter 1

Introduction

The aim of this project is to investigate methods to predict the population level of towns and cities during the intermediate years between censuses, which are conducted every ten years.

The population level of Chester is important data to the Chester City Council as it is used by the government to determine the money to be made available for the council to provide its public services.

The population levels should be very accurate for the census years, the most recent being 1981 and 1991, and at the next census in 2001, but between these years the government needs to estimate the totals. The government's method of estimating the intercensal population has been modified frequently over the years, and the most recent methods will be discussed shortly.

The amount of money made available is calculated as the Standard Spending Assessment (SSA), which is calculated per head of population. This means that any underestimate in the population will result in a shortfall in government funds for Chester City Council to spend on their local services. The amount of money made available by the government to all local authorities is a fixed sum, and so one authority's loss is another's gain.

A detailed description of the SSA is contained in the appendix.

The government frequently modifies its method of estimating the intercensal population. These are described in detail from 1854 in the Occasional Paper 37 [6] but we will only summarise the later years. From 1954 to 1971 the method looked at two sets of figures. The first used figures for migration derived from electoral registers and the second looked at migration figures supplied by each local authority using changes in housing stock. If these two annual estimates differed then a weighted average was used, with the housing element having twice the weight of the electoral element. This figure was then added onto the current running total estimate.

Results of the 1961 Census of Population showed that the difference between the pop-

ulation estimates carried forward from the 1951 census and the 1961 census figures was generally under 3 per cent.

However, the results carried forward from 1961 to the 1971 census showed much larger discrepancies, some up to 10 per cent. As a consequence the methods were reviewed. The housing returns were abandoned as these were found to be less reliable.

After the 1971 Census a new source was considered for obtaining data on migration levels. This was a ten per cent sample of re-registrations notified to the National Health Service Central Register. Also a feasibility study examined if the scope for the annual canvass for the electoral register should be extended. This latter idea was not implemented after the study because of the cost and the extra burden on the public.

1974 saw a major change with the number of local authorities reduced from 1400 to 400. The increase in computer power enabled the local health authority to compile data by age and sex, improving local estimates.

The 1981 Census results were also compared with the carried forward estimates from 1971 to 1981. Seventy per cent of estimates were within 2.5 per cent of the census figures, with the average error being 2 per cent below. The maximum errors were under and overestimates as high as 12 per cent.

Since 1981 the main change has been the increase from 10 per cent to 100 per cent use of the National Health Service Central Register.

Government population estimates methodology. Mid year population estimates are currently made for about 500 'building bricks' within England and Wales. The formula used is

$$\begin{aligned} P_{n+1} = \; & PA_n + B_n - D_n \\ & + CIA_n - CEA_n \\ & + CII_n - CEI_n \\ & + SAF_{n+1} \\ & + FAFD_{n+1} \\ & + PR_{n+1} \\ & + TTS_{n+1} \end{aligned}$$

where

P_{n+1}	=	resident population mid year estimate $n+1$ years after last census
PA_n	=	adjusted aged-on estimate of the building brick at mid year n
B_n	=	births in the twelve months in year n to mothers usually resident in the building brick
D_n	=	deaths in the twelve months in year n of people usually resident in the building brick
CIA_n	=	civilian immigrants into the building brick from outside the UK in year n
CEA_n	=	civilian emigrants from the building brick to outside the UK in year n
CII_n	=	civilian immigrants into the building brick from elsewhere in the UK in year n
CEI_n	=	civilian emigrants from the building brick to elsewhere in the UK in year n
SAF_{n+1}	=	foreign and UK armed forces personnel stationed in the building brick at mid year $n+1$
$FAFD_{n+1}$	=	foreign armed forces dependents resident in the building block at mid year $n+1$
PR_{n+1}	=	prisoners in prisons in the building brick at mid year $n+1$
TTS_{n+1}	=	term time students resident in the building brick (in the base year), plus number of children resident in boarding schools in the building brick at mid year $n+1$

Discussion. The method used has an obvious problem. If incorrect data is fed into the equation in a year soon after the census, then that error will remain in the system until the next census becomes the base year. Clearly, a couple of early underestimated years could result in a large loss of revenue for the council for up to ten years. Even if the new census puts the population back to the correct level, the loss of revenue in the previous period is irretrievable.

A further problem is that it takes up to two years to analyse the census data which means that the base of 1981 is incremented until 1993, which gives further time for errors to propogate.

At this stage it is interesting to look at the census figures and intercensal estimates for Chester to see if there are grounds for concern. The data is illustrated as figure 1.1.

It appears from this graph that in the period 1981-1991 Chester's population was underestimated, and hence the city received a shortfall government money. This was especially noticable during the final years with the large increase from 1990 to the census level of 1991. A simple linear representation between the census years shows the estimated levels below this line for the complete period. The concern of accumulation of errors

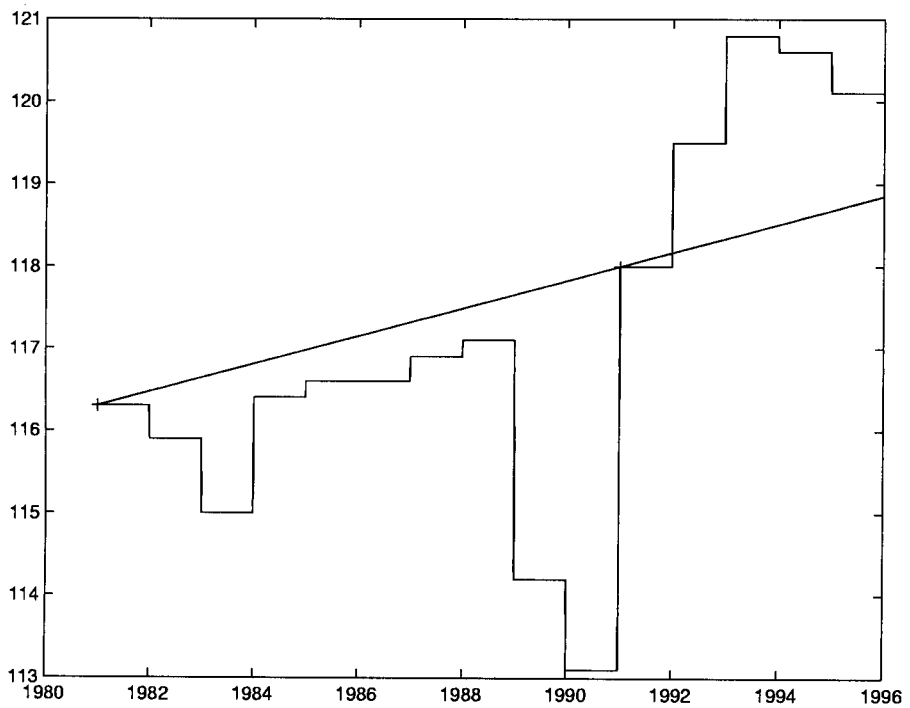

Figure 1.1: Government mid year estimates for Chester in 1000s

suggested in the previous section seems to have been borne out. The situation in the current period is more advantageous towards Chester, but this nowhere near redresses the previous losses and the level is once again falling towards the linear trend, indicating that by the 2001 Census the shortfall could recur.

With local services depending upon the SSA, which in turn depends upon accurate population data in the intercensal years, it is the aim of this project to investigate the estimates of the intercensal figures.

The Approach There are two clear approaches that can be followed. The first is to follow through the government's methodology and to check the data used to update the levels year by year. This would require a huge amount of resources to execute, both in time and money, and is well outside the scope for this project. The second is to attempt to model the accurate census data to predict mathematically future levels. We will follow the second approach in a structured way, using interpolation and extrapolation techniques and more specific methods as outlined below.

In the following chapter we will discuss the theory of standard interpolation techniques and other approximation methods. These techniques will then be tested on a simple smooth function which will be chosen to look similar to a gradually increasing population. Methods which clearly do not produce a good approximation to the function will be discarded. The

following chapter will introduce actual census data and a standard population model will be tested using some of this data.

We will then continue by applying the more successful methods to some of the census data to compare the fit. The remaining data can finally be used to see if we can predict the 1991 population level from the census data for 1951 - 1981.

This procedure will enable us to decide if it is possible to use mathematical methods to predict the next set of census data and, if this is possible, the intercensal figures will follow easily.

Conclusion As a result of this investigation we conclude that it is not possible to predict population levels mathematically . We show that the population figures cannot be made to fit standard interpolating functions and that the figures lack the uniformity necessary for neural networks to recognise regular patterns.

It will be necessary to check the accuracy of the data obtained and used in the government model in order to confirm that the intercensal data for Chester is an underestimate, and that the council is suffering from a shortfall in funding.

Chapter 2

Interpolation Methods

This chapter will consider various methods of interpolation. In each case the method will fit a function to the n data points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. Also we define $h_i = x_{i+1} - x_i$, where the h_i are not necessarily equal.

2.1 Polynomial Methods

2.1.1 General Polynomials

We can fit any number of polynomials of the form

$$y_m(x) = a_m x^m + a_{m-1} x^{m-1} + \cdots + a_1 x + a_0; \qquad m \geq n - 1 \tag{2.1}$$

In addition, if $a_i = 0$ for $m - n + 1$ values of i then this polynomial will be unique. For example, taking $n = 3$, $m = 5$ we need 3 zero coefficients in a unique interpolating polynomial $y_5(x)$. There are clearly many forms that can be used,

$$y_5(x) = a_5 x^5 + a_4 x^4 + a_0$$

and

$$y_5(x) = a_5 x^5 + a_3 x^3 + a_1 x$$

are two such forms. In each case, substituting for $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ will give three linear equations for the three non zero coefficients, which can usually be solved to find the interpolating polynomial for $y_5(x)$. Without the condition of three zero coefficients a general interpolating polynomial $y_5(x) = \sum_{i=0}^{5} a_i x^i$ will result in three linear equations in the six coefficients $a_0, \ldots, a_5$, and clearly we would not expect a unique solution.

Various forms of this polynomial will be investigated in chapter 3, using MATLAB for graphical comparisons.

2.1.2 Lagrangian Interpolation

This standard technique, which can be found in any text on interpolation such as Evans [3, pages 54 - 56], avoids the need to solve a set of linear equations when finding the interpolating polynomial.

We begin by defining the function

$$l_k(x) = \frac{(x - x_1) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n)}{(x_k - x_1) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n)} \tag{2.2}$$

This function can be simplified by defining a second function

$$\prod(x) = (x - x_1)(x - x_2) \cdots (x - x_{n-1})(x - x_n) \tag{2.3}$$

and hence

$$l_k(x) = \frac{\prod(x)}{(x - x_k)\prod'(x_k)} \tag{2.4}$$

It is clear from equation (2.2) that $l_k(x)$ takes the value 0 for all $x = x_i$ except $x = x_k$ for which $l_k(x_k) = 1$.

If we now define

$$y_n(x) = \sum_{k=1}^{n} l_k(x) y_k \tag{2.5}$$

then we clearly have a polynomial of degree n for which $y_n(x_k) = y_k$ for $k = 1, \ldots, n$

Equation (2.5) defines the Lagrangian interpolation function, and in chapter 3 we will show that this is the polynomial obtained in section 2.1.1 with $m = n - 1$.

2.1.3 Hermite Interpolation

The Lagrangian interpolation function agrees with the data at the n data points. A more accurate interpolator would be found if the gradient y'_k was given at the n tabulated values. We would then need to find a function $y_n(x)$ such that $y'_n(x_k) = y'_k$ in addition to $y_n(x_k) = y_k$. The technique for finding such a function is fully covered in [3, pages 57 - 59]. As in this project we will be dealing with discrete data only with no means of obtaining gradients, this method will not be feasible for fitting the population data and will be discussed no further.

2.1.4 Divided Differences Methods

We can define recursively the divided difference of order $k > 1$ of $y(x)$ by

$$y[x_1, x_2, \ldots, x_k] = \frac{y[x_2, \ldots, x_k] - y[x_1, \ldots, x_{k-1}]}{(x_k - x_1)} \tag{2.6}$$

with $y[x_i] = y_i$, as quoted in Ralston and Rabinowitz [5, page 85].

Divided differences can be calculated neatly in a table as follows:-

For $k > i \geq 1$ we have the general form

$$y[x_i, x_{i+1}, \ldots, x_k] = \frac{y[x_{i+1}, \ldots, x_k] - y[x_i, \ldots, x_{k-1}]}{(x_k - x_i)}$$

$$\begin{array}{llll}
x_1 & y_1 & & \\
 & & y[x_1, x_2] & \\
x_2 & y_2 & & y[x_1, x_2, x_3] \\
 & & y[x_2, x_3] & & y[x_1, x_2, x_3, x_4] \\
x_3 & y_3 & & y[x_2, x_3, x_4] \\
 & & y[x_3, x_4] & \\
x_4 & y_4 & &
\end{array} \tag{2.7}$$

Each additional column to the right is calculated by subtracting the adjacent column values and dividing by the required x difference as defined by the general form above.

[5] states, without proof, that for $n \geq 2$

$$y[x_1, x_2, \ldots, x_n] = \sum_{i=1}^{n} \frac{y(x_i)}{(x_i - x_1)(x_i - x_2)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_n)} \tag{2.8}$$

This can be proved by induction as follows:-

Try $n = 2$,

$$\begin{aligned}
y[x_1, x_2] &= \frac{y(x_2) - y(x_1)}{(x_2 - x_1)} \\
&= \frac{y(x_1)}{(x_1 - x_2)} + \frac{y(x_2)}{(x_2 - x_1)}
\end{aligned}$$

which is of the form of the RHS of equation (2.8) with $n = 2$.

Now assume that (2.8) is true for $n = k$. That is, assume

$$y[x_1, x_2, \ldots, x_k] = \sum_{i=1}^{k} \frac{y(x_i)}{(x_i - x_1)(x_i - x_2)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_k)} \tag{2.9}$$

Now try $n = k + 1$ The LHS of equation (2.8) is

$$
\begin{aligned}
y[x_1, x_2, \ldots, x_{k+1}] &= \frac{y[x_2, \ldots, x_{k+1}] - y[x_1, \ldots, x_k]}{(x_{k+1} - x_1)} \\
&= \left[\sum_{i=2}^{k+1} \frac{y(x_i)}{(x_i - x_2) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_{k+1})} \right] / (x_{k+1} - x_1) \\
&\quad - \left[\sum_{i=1}^{k} \frac{y(x_i)}{(x_i - x_1) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_k)} \right] / (x_{k+1} - x_1)
\end{aligned}
\tag{2.10}
$$

Now for $i = 1$ we only have one term in the second expression of the RHS of equation (2.10) and this is

$$
-\frac{y(x_1)}{(x_1 - x_2) \ldots (x_1 - x_k)(x_{k+1} - x_1)} = \frac{y(x_1)}{(x_1 - x_2) \ldots (x_1 - x_k)(x_1 - x_{k+1})}
$$

which is of the correct form. Also for $i = k+1$ we only have one term in the first expression of the RHS of equation (2.10) and this is

$$
\frac{y(x_{k+1})}{(x_{k+1} - x_1) \ldots (x_{k+1} - x_k)}
$$

which is also of the correct form.

For $2 \leq i \leq k$ the RHS of equation (2.10) becomes

$$
\begin{aligned}
RHS &= \frac{y(x_i)}{(x_i - x_2) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_{k+1})(x_{k+1} - x_1)} \\
&\quad - \frac{y(x_i)}{(x_i - x_1) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_k)(x_{k+1} - x_1)} \\
&= \frac{((x_i - x_1) - (x_i - x_{k+1}))y(x_i)}{(x_i - x_1) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_{k+1})(x_{k+1} - x_1)} \\
&= \frac{(x_{k+1} - x_1)y(x_i)}{(x_i - x_1) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_{k+1})(x_{k+1} - x_1)} \\
&= \frac{(y(x_i)}{(x_i - x_1) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_{k+1})}
\end{aligned}
$$

which is also of the desired form. This concludes the proof by induction.

The result (2.10) indicates that the order of the arguments in the divided difference is not important. Hence

$$
\begin{aligned}
y[x_1, x_2, \ldots, x_k, x] &= y[x, x_1, x_2, \ldots, x_k] \\
&= \frac{y[x_1, \ldots, x_k] - y[x, x_1, \ldots, x_{k-1}]}{(x_k - x)}
\end{aligned}
$$

This gives the recurrence relation

$$y[x_1, x_2, \ldots, x_{k-1}, x] = y[x_1, x_2, \ldots, x_k] + (x - x_k)y[x_1, x_2, \ldots, x_k, x] \tag{2.11}$$

Repeated use of equation (2.11) produces the formula

$$\begin{aligned} y(x) &= y_1 + (x - x_1)y[x_1, x_2] + (x - x_1)(x - x_2)y[x_1, x_2, x_3] \\ &\quad + \cdots + (x - x_1)\cdots(x - x_{k-1})y[x_1, x_2, \ldots, x_k] + \cdots \\ &\quad + (x - x_1)\cdots(x - x_k)y[x_1, \ldots, x_k, x] \end{aligned} \tag{2.12}$$

[5, page 85] shows that the final term tends to zero as k increases and equation (2.12) is known as Newton's divided difference interpolation formula. Chapter 3 will demonstrate that this also gives the polynomial obtained in section 2.1.1 with $m = n - 1$.

There are a number of other divided difference formulae, such as those by Stirling and by Everett for example.

2.1.5 Neville's Algorithm

This is a very practical algorithm for finding interpolating polynomials and is described in Harding and Quinney [4, pages 27-31]. The method uses a subset of the tabulated points to construct the interpolant and a simple algorithm is used to add an additional point to produce a better interpolant with the degree increased by one. In practice we can continue to add points until the interpolating polynomial shows close agreement. The algorithm is particularly useful in this work on population data as it is now simple to add a later set of census data without the need to recalculate the whole process.

Construct the linear functions:-

$$\begin{aligned} p_{1,2}(x) &= \frac{(x - x_1)y_2 - (x - x_2)y_1}{(x_2 - x_1)} \\ p_{2,3}(x) &= \frac{(x - x_2)y_3 - (x - x_3)y_2}{(x_3 - x_2)} \\ \ddots &= \ddots\ddots\ddots\ddots\ddots\ddots\ddots\ddots \\ p_{n-1,n}(x) &= \frac{(x - x_{n-1})y_n - (x - x_n)y_{n-1}}{(x_n - x_{n-1})} \end{aligned}$$

These terms are called the linear cross means. Clearly $p_{j,j+1}(x)$ passes through the points (x_j, y_j) and (x_{j+1}, y_{j+1}) for $j = 1, 2, \ldots, n - 1$. We can now define the quadratic cross mean between two successive linear functions by

$$p_{j,j+2}(x) = \frac{(x - x_j)p_{j+1,j+2} - (x - x_{j+2})p_{j,j+1}}{(x_{j+2} - x_j)}$$

This quadratic clearly passes through the points $(x_j, y_j), (x_{j+1}, y_{j+1}), (x_{j+2}, y_{j+2})$ for

$j = 1, 2, \ldots, n-2$.
Finally if we define recursively

$$p_{j,k}(x) = \frac{(x - x_j)p_{j+1,k} - (x - x_k)p_{j,k-1}}{(x_k - x_j)} \tag{2.13}$$

for $k > j, k = 2, 3, \ldots, n$. This is a polynomial of degree $(k - j)$ passing through all the points $(x_j, y_j), \ldots, (x_k, y_k)$. The polynomial we need is $p_{1,n}(x)$ and chapter 3 will once again show that $p_{1,n}(x)$ is the polynomial obtained in section 2.1.1. It is now just a simple process to find the polynomial $p_{1,n+1}$ through an additional point (x_{n+1}, y_{n+1}) without repeating significant amounts of algebra.

2.2 Rational Function Interpolation

2.2.1 General Rational Functions

The disadvantage of using polynomials for approximations is their tendency for oscillation [1, page 459]. Error bounds may then exceed the average approximation error as error bounds are determined by the maximum approximation error. A technique which spreads the approximation error more evenly uses rational functions. We define a rational function $r(x)$ of degree m by $r(x) = \frac{p(x)}{q(x)}$ where $p(x), q(x)$ are polynomials whose degrees sum to m and the first term in $q(x)$ has unit coefficient. Error bounds using rational functions will be no more than for polynomials, as taking $q(x) = 1$ shows that any polynomial is also a rational function.

With reference to section 2.1.1 we once again need $m \geq n - 1$ and for $r(x)$ to be determined uniquely for our n tabulated points we need a total of $m - n + 1$ coefficients in $p(x)$ and $q(x)$ to be zero.

Again, if $n = 3$ and $m = 5$ we need 3 zero coefficients. Many forms can be used, with

$$r_5(x) = \frac{p_3x^3 + p_2x^2}{x^2 + q_0}; \qquad p_1 = p_0 = q_1 = 0$$

and

$$r_5(x) = \frac{p_2x^2}{x^3 + q_1x + q_0}; \qquad p_1 = p_0 = q_2 = 0$$

being two examples. In each case, substituting for $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ will give three linear equations for the three non zero coefficients, which can usually be solved to find the interpolating rational for $r_5(x)$. Without the condition of three zero coefficients a general

interpolating rational such as $r_5(x) = \dfrac{p_3x^3 + p_2x^2 + p_1x + p_0}{x^2 + q_1x + q_0}$ will result in three linear equations in the six coefficients $p_0, \ldots, p_3, q_0, q_1$, and clearly we would not expect a unique solution.

Various forms of rational interpolation will be investigated in chapter 3, using MATLAB for graphical comparisons.

2.2.2 Padé Approximations

Rational approximations $r(x)$ to a function $f(x)$ can be made such that $r^{(k)}(0) = f^{(k)}(0)$ for $k = 1, \ldots, m$. This is called the Padé approximation. However, as this project is to fit tabular points, and derivatives are not available, this technique will not be considered any further.

2.3 Cubic Splines

This technique fits a cubic function across each of the $(n-1)$ intervals $[x_i, x_{i+1}]$ of the form

$$f_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{2.14}$$

for $i = 1, 2, \ldots, n-1$. With (n-1) functions we need to find values for the $4(n-1)$ constants a_i, b_i, c_i, d_i. These constants can be calculated from the following conditions:-

1. The cubics pass through the tabulated points.
2. The first derivatives are continuous at the tabulated points $x_2, \ldots, x_{n-1}$.
3. The second derivatives are continuous at the tabulated points $x_2, \ldots, x_{n-1}$.
   This gives us $2(n-1) + (n-2) + (n-2) = 4(n-1) - 2$ conditions. This leaves 2 more conditions needed and boundary conditions at x_1 andx_n such as
4. The second derivatives at x_1 and x_n are zero

can be used. In Mathematical terms these conditions reduce to

1. $f_i(x_i) = y_i$ and $f_i(x_{i+1}) = y_{i+1}$; $\quad i = 1, 2, \ldots, n-1$
2. $f_i'(x_{i+1}) = f_{i+1}'(x_{i+1})$; $\quad i = 1, 2, \ldots, n-2$
3. $f_i''(x_{i+1}) = f_{i+1}''(x_{i+1})$; $\quad i = 1, 2, \ldots, n-2$
4. $f_1''(x_1) = 0 = f_{n-1}''(x_n)$

The method of Burden and Faires [1, pages 126-132] will be discussed in this section as it is simple to use with linear algebra software. Condition 1 above gives us the equations

$$a_i = y_i; \qquad i = 1, \ldots, n-1 \tag{2.15}$$

and using $h_i = x_{i+1} - x_i$ as usual

$$a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = y_{i+1}$$

which together with equation (2.15) leads to

$$b_i h_i + c_i h_i^2 + d_i h_i^3 = y_{i+1} - y_i; \qquad i = 1, \ldots, n-1 \tag{2.16}$$

We know that $f_i'(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2$ and hence condition 2 gives us

$$b_i + 2c_i h_i + 3d_i h_i^2 = b_{i+1}; \qquad i = 1, \ldots, n-2 \tag{2.17}$$

and also $f_i''(x) = 2c_i + 6d_i(x - x_i)$ so condition 3 gives us

$$c_i + 3d_i h_i = c_{i+1}; \qquad i = 1, \ldots, n-2 \tag{2.18}$$

Condition 4 finally gives us 2 more equations. $f_1''(x_1) = 0$ implies that

$$c_1 = 0 \tag{2.19}$$

and [1] neatly considers any cubic spline to the right of x_n. For this cubic the condition $f_{n-1}''(x_n) = 0$ together with condition 3 gives $f_n''(x_n) = 0$.This implies that

$$c_n = 0 \tag{2.20}$$

Although $f_n(x)$ is not required this idea simplifies the algebra and [1] proceeds to produce a set of n linear equations for the constants c_i, $i = 1, \ldots, n$ as follows:-

Substituting for d_i from equation (2.18) into equation (2.17) gives

$$b_i + h_i(c_i + c_{i+1}) = b_{i+1}; \qquad i = 1, \ldots, n-2 \tag{2.21}$$

and substituting for d_i from equation (2.18) into equation (2.16) produces

$$b_i h_i + c_i h_i^2 + \frac{h_i^2}{3}(c_{i+1} - c_i) = y_{i+1} - y_i$$

which simplifies to

$$b_i + \frac{h_i}{3}(2c_i + c_{i+1}) = \frac{(y_{i+1} - y_i)}{h_i} \tag{2.22}$$

If we now increase the index of equation (2.22) by 1 we get

$$b_{i+1} + \frac{h_{i+1}}{3}(2c_{i+1} + c_{i+2}) = \frac{(y_{i+2} - y_{i+1})}{h_{i+1}}; \qquad i = 0, \ldots, n-1 \tag{2.23}$$

We can now substitute for b_i, b_{i+1} from equations (2.22), (2.23) into (2.21) and simplify to give

$$h_i c_i + 2(h_i + h_{i+1})c_{i+1} + h_{i+1}c_{i+2} = \frac{3}{h_{i+1}}(y_{i+2} - y_{i+1}) - \frac{3}{h_i}(y_{i+1} - y_i) \quad (2.24)$$
$$i = 1, 2, \ldots, n-2$$

and these equations, together with $c_1 = 0$ and $c_n = 0$ from the boundary conditions, give the n equations for finding $c_i, \quad i = 1, \ldots, n$

The equations can be expressed in matrix form as

$$A\underline{x} = \underline{b} \quad (2.25)$$

where

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \cdots & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & 0 & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & 0 & 0 & 1 \end{bmatrix}$$

$$\underline{x} = [c_1 \cdots c_n]^T$$

$$\underline{b} = \begin{bmatrix} 0 \\ \frac{3}{h_2}(y_3 - y_2) - \frac{3}{h_1}(y_2 - y_1) \\ \vdots \\ \frac{3}{h_{n-1}}(y_n - y_{n-1}) - \frac{3}{h_{n-2}}(y_{n-1} - y_{n-2}) \\ 0 \end{bmatrix}$$

As matrix A is strictly diagonally dominant a unique solution for $\underline{x}$, or c_i can be found. It is then a simple process to substitute into equations (2.22), (2.18) to find b_i, d_i and hence each cubic spline. This technique will be illustrated in chapter 3.

2.4 Least Squares Method

In section 2.1.1 we considered fitting polynomials of degree $m \geq n-1$ through our tabulated points. However, it is possible to fit an approximating polynomial of degree $m < n-1$ by

minimising the sum of the squared vertical differences between the tabulated points and the polynomial. If we define our approximating polynomial by $p_m(x) = \sum_{i=0}^{m} a_i x^i$ then we clearly need to minimise the expression

$$E = \sum_{i=0}^{m} (y_i - p_m(x_i))^2 \tag{2.26}$$

To find a_i, $i = 0, 1 \ldots, m$ to minimise E we need to solve the $m + 1$ equations given by

$$\frac{\partial E}{\partial a_i} = 0 \tag{2.27}$$

This method will also be demonstrated in chapter 3

2.5 Neural Networks

Neural Networks are composed of simple elements operating in parallel [2, page 1-2]. These elements are analogous to the neurons in the brain and are interconnected. The connections of the artificial neurons are called weights, and the network can be trained to perform different tasks by adjusting these connecting weights.

During training the network is given target output values corresponding to particular input vectors. Initially the weights are randomly selected and the network produces an output. This output is compared with the target and the error is computed. Different routines, such as 'gradient descent', are applied to modify the weights in order to reduce this error. This process is repeated until the target output is attained or the error is within acceptable tolerances. A simple flow chart, figure 2.1, shows the process of training.

The network can now be given new input vectors and it computes an output corresponding to the patterns observed in training, incorporated in the final weights.

The networks used in this project are all set up and programmed in the MATLAB Neural Network Toolbox and are described in its User Guide [2].

The programs consist of a hidden layer of a varying number, n, of neurons, each receiving a weighted input from the input vectors. Each neuron calculates its input value as the scalar product of the input vector and its associated weight vector. This value is then applied to a transfer function to calculate its output value. Each hidden neuron in this layer uses the same transfer function. The n output values from the first layer form another vector and this is combined as another scalar product with the second layer weight vector to produce the input to the second and output neuron. This weighted sum is then applied to another transfer function to give the network output. The flowchart, figure 2.3, shows a typical set up with a hidden layer of 3 neurons.

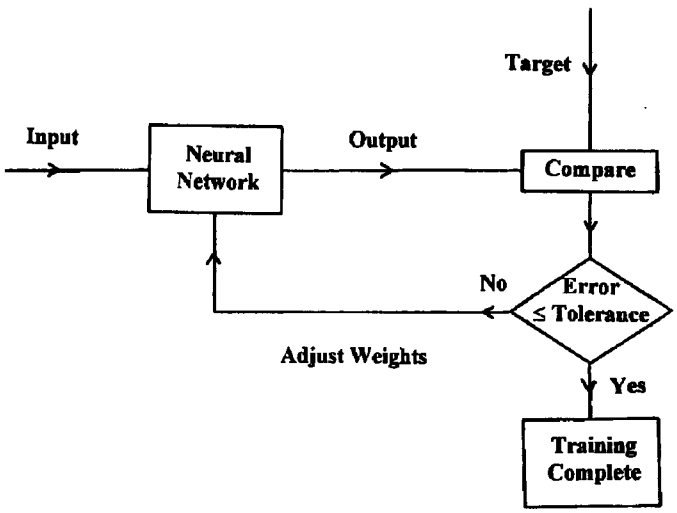

Figure 2.1: Neural Network Training Flowchart.

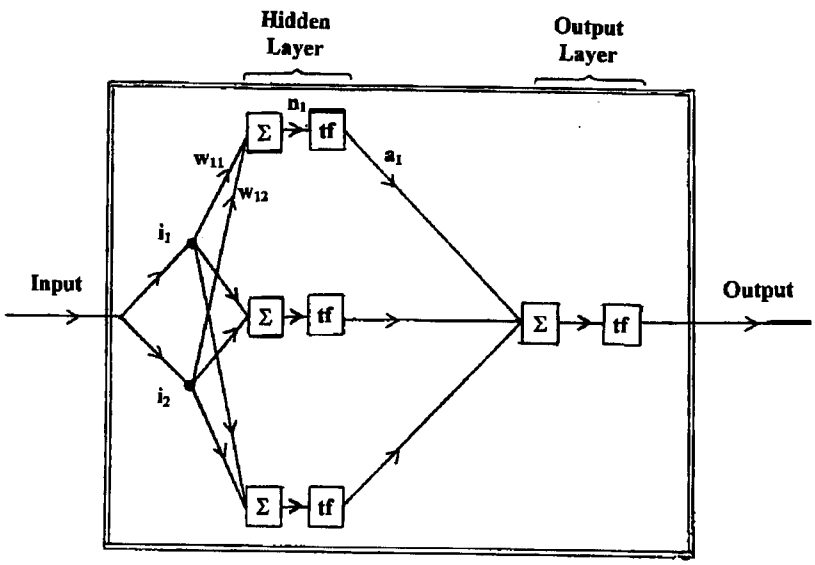

Figure 2.2: Network with 3 hidden neurons in inner layer.

For this network

$$n_1 = w_{11}i_1 + w_{12}i_2 \tag{2.28}$$

$$a_1 = tf(n_1) \tag{2.29}$$

and n_2, n_3, a_2 and a_3 can be calculated in a similar form. The input and output for the second layer neuron is also calculated in a similar form.

There are a number of standard transfer functions. Typical functions 'hard limit', 'pure linear' and 'sigmoid' are shown in figure 2.3, but many more exist.

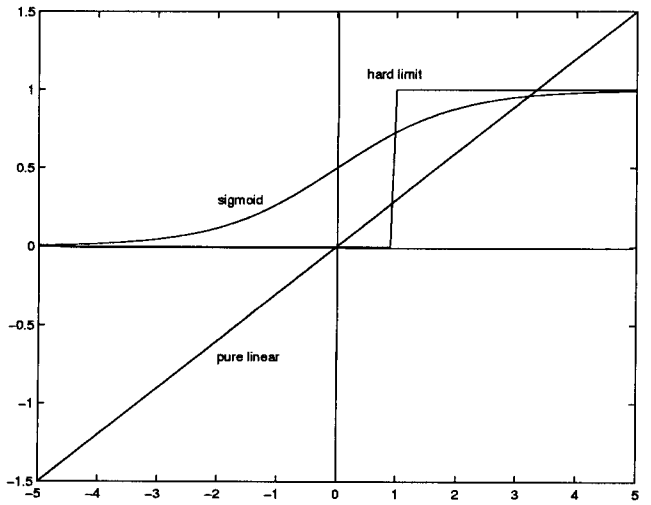

Figure 2.3: Neural network transfer functions.

The training and stopping techniques used in this project are all described in detail in the User Guide [2].

One method used is 'gradient descent', which is used in the MATLAB routine 'traingd'. The error function, e, which is often referred to as the energy of the network, is defined by

$$e = \frac{1}{2}\sum_{i=1}^{m} |o_i - t_i|^2 \tag{2.30}$$

where o_i = is the output for training vector i

and t_i is the target for training vector i.

The weights are adjusted to move the error function down its tangent at each step until it reaches a minimum. which may be a local minimum. An improved method of 'gradient descent with momentum', used as 'traingdx' by MATLAB, moves the method through a local minimum to reach a global minimum.

The Automated Regularization method, used as 'traingbr' by MATLAB, uses a Bayesian framework. The weights are assumed to be random variables and statistical techniques are used to estimate their parameters.

Early Stopping is another technique used in this project. This training routine is used to avoid overfitting, a problem described in chapter 5. Early stopping divides the training data into three subsets. The first subset acts as a training set as described above, using any of the training routines. The second subset is used to validate the process, and the

validation error will reduce initially. However, if the network is starting to overfit the data the validation error will start to increase. When this increase begins the training will stop 'early'. The third subset is not used in training, but is used to compare different models.

There are many other MATLAB routines.

2.6 Logistic Curve

Many populations naturally follow a model which is the solution to the logistic differential equation $\frac{dP}{dt} = aP(1 - P/M)$ where P is the population at time t and a, M are constants. The General Solution of this differential equation can easily be found by the method of separation of variables in texts such as [8, page 46] and can be shown to be

$$P = \frac{M}{1 + Ke^{-at}} \tag{2.31}$$

where constant $K = M/P_0 - 1$ and P_0 is the initial population. If the population is known at regular time intervals (as in a census) then it is straightforward to check if the data appears to follow this logistic model. Let P_n be the population at time t_n where $t_n = nt$ and t is the constant time interval between data readings. Equation (2.31) gives $P_{n+1} = \frac{M}{1 + Ke^{-a(n+1)t}}$ and $P_n = \frac{M}{1 + Ke^{-ant}}$ and, by eliminating K and rearranging these expressions, we get

$$\frac{P_{n+1}}{P_n} = \frac{(1 - e^{at})}{M} P_{n+1} + e^{at} \tag{2.32}$$

and hence a graph of $\frac{P_{n+1}}{P_n}$ against P_{n+1} will produce a straight line if the model is appropriate, and the constants a, M can be estimated from this graph.

This method will not be tested on the general function chosen in chapter 3 as the function is not population data. However, it will be used when testing the sets of census data from 1951-1991.

Chapter 3

Testing Interpolation Methods

The function chosen to test the methods clearly needs to be non-polynomial else the polynomial methods would be exact fits. It also needs to have some similarities to the shape one might expect for population data. For some of the methods it needs to be easily differentiable. The function selected is $y = \sqrt[3]{x}$ through the four tabulated points $(0,0),(1,1),(8,2),(27,3)$. The graph below, figure 3.1, shows this function and the tabulated points. It can be seen to be increasing, but more slowly towards the later values which mirrors a population with a decreasing birth rate and this seems a sensible scenario.

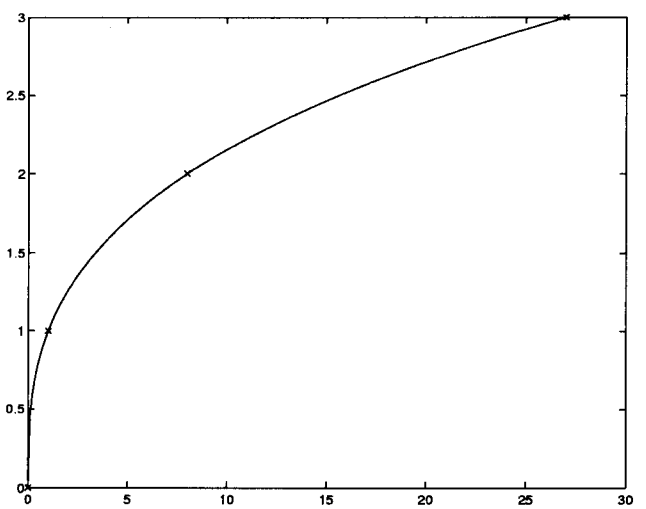

Figure 3.1: Test function, $y = \sqrt[3]{x}$

3.1 Polynomial Methods

3.1.1 General Polynomials

With reference to section 2.1.1 we will fit

$$y_5(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_0 \tag{3.1}$$

and

$$y_5(x) = a_4x^4 + a_2x^2 + a_1x + a_0 \tag{3.2}$$

and finally the standard polynomial for 4 tabulated points

$$y_3(x) = a_3x^3 + a_2x^2 + a_1x + a_0 \tag{3.3}$$

which is of degree 4-1=3 with no zero coefficients in its general form.

Note that for functions (3.1) and (3.2) the tabulated point $(0, 0)$ will always give $a_0 = 0$ and so in this case we only have three non zero coefficients in a unique interpolating polynomial

Substituting the four tabulated points into equation (3.1) will give the following four linear equations

$$\begin{array}{rrrrcl} 531441a_4 & +19683a_3 & +729a_2 & +a_0 & = & 3 \\ 4096a_4 & +512a_3 & +64a_2 & +a_0 & = & 2 \\ a_4 & +a_3 & +a_2 & +a_0 & = & 1 \\ & & & a_0 & = & 0 \end{array} \tag{3.4}$$

Solving these equations and substituting into equation (3.1) gives the interpolating polynomial

$$y_4(x) = 0.005267874x^4 - 0.1858037x^3 + 1.1805358x^2 \tag{3.5}$$

The graph of function (3.5) is drawn, figure 3.2, with our test function and although it passes through the tabulated points it is clearly a very poor fit.

We can repeat the process to find the interpolating polynomial for (3.2), which produces

$$y_4(x) = 0.0001066591x^4 - 0.1149290x^2 + 1.1148223x \tag{3.6}$$

and finally the standard polynomial for 4 points (3.3) which gives

$$y_3(x) = 0.00383972x^3 - 0.1417004x^2 + 1.1378606x \tag{3.7}$$

To complete this section we can plot the final two calculated polynomials, figure 3.3, which are much better fits than the first, together with our original function for comparison.

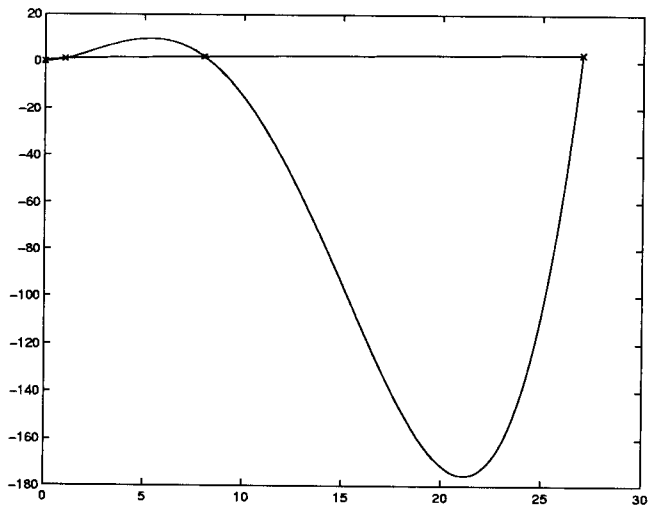

Figure 3.2: Graph of function (3.5)

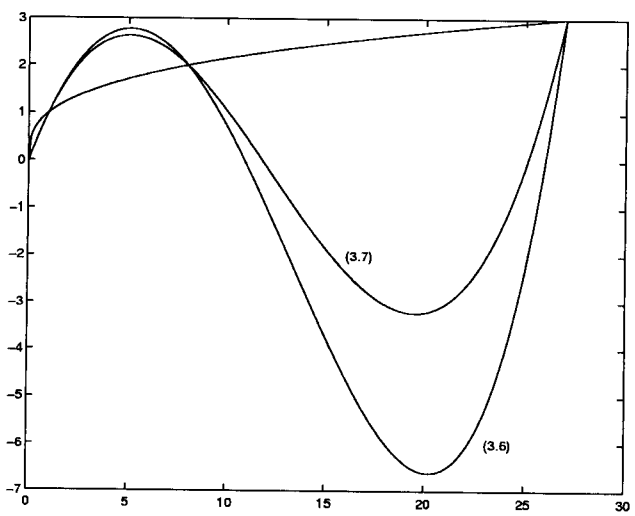

Figure 3.3: Graph of functions (3.6) and (3.7)

The standard polynomial is clearly the best fit. However, this fit is far from being acceptable and immediately puts into question the use of a polynomial function to interpret our population data.

3.1.2 Lagrangian Interpolation

Using the method of section 2.1.2 the Lagrangian interpolation polynomial is

$$\begin{aligned} y(x) &= \frac{(x-1)(x-8)(x-27)}{(-1)(-8)(-27)}.0 + \frac{x(x-8)(x-27)}{(1)(-7)(-26)}.1 \\ &= +\frac{x(x-1)(x-27)}{(8)(7)(-19)}.2 + \frac{x(x-1)(x-8)}{(27)(26)(19)}.3 \end{aligned} \tag{3.8}$$

which can be simplified to

$$y(x) = 0.00383972x^3 - 0.1417004x^2 + 1.1378606x$$

which is identical to the function calculated in (3.7) as one would expect. This function has already been shown to be a poor fit.

3.1.3 Hermite Interpolation

As discussed in section 2.1.3 this method uses the derivatives at the tabulated points and so is not suitable when only data points are available. However, it is interesting to compare the Lagrangian with the Hermite for our test function. Using the method of [3, pages 57-59], and choosing our polynomial to have the same gradient as our test function at the points $(1,1),(8,2)$ only, the Hermite polynomial can be shown to be

$$\begin{aligned} H(x) &= \frac{x(x-1)^2(x-8)^2}{2196324} + \frac{x(123x-305)(x-8)^2(x-27)}{231868} \\ &= +\frac{x(381x-4112)(x-1)^2(x-27)}{3962336} - \frac{x(x-1)(x-8)^2(x-27)}{3822} \\ &= -\frac{x(x-1)^2(x-8)(x-27)}{89379} \end{aligned}$$

We can now plot this with the Lagrangian and the test function as figure 3.4

From the graph it is clear that even with the extra information the Hermite polynomial is a poorer fit. If we reduce the range of x values to examine the more successful part of the fit we can see, in figure 3.5, that the Hermite is still no better.

The problem is that the Hermite polynomial is of degree 5 and section 3.1.1 demonstrated that higher degree polynomials generally produced poorer fits.

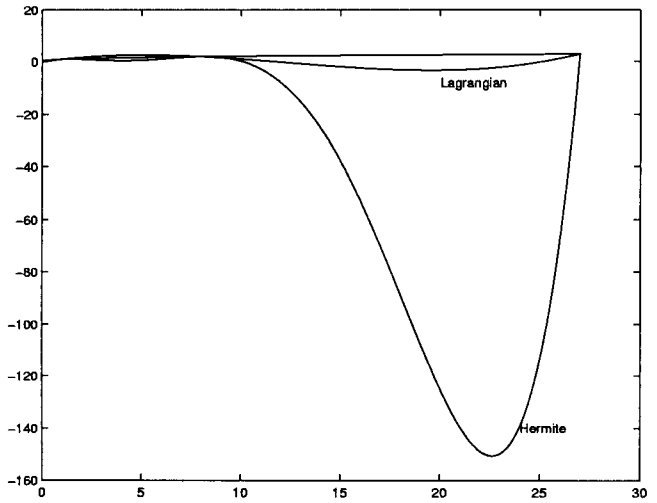

Figure 3.4: Lagrangian and Hermite Interpolating Polynomials

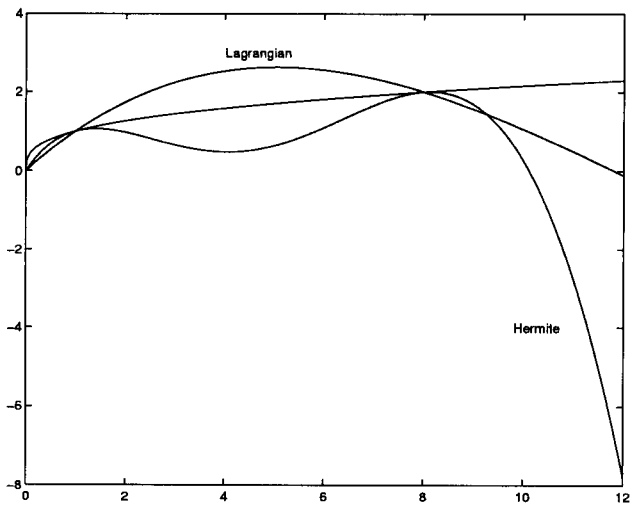

Figure 3.5: Reduced Domain Lagrangian and Hermite Interpolating Polynomials

3.1.4 Divided Differences

We defined in section 2.1.4 the recursive formula (2.6)

$$y[x_1, x_2, \ldots, x_k] = \frac{y[x_2, \ldots, x_k] - y[x_1, \ldots, x_{k-1}]}{(x_k - x_1)}$$

with $y[x_i] = y_i$.

We can now draw up the divided difference table

$$\begin{array}{cccccc} 0 & 0 & & & & \\ & & 1 & & & \\ 1 & 1 & & -\frac{3}{28} & & \\ & & \frac{1}{7} & & \frac{239}{62244} & \\ 8 & 2 & & -\frac{6}{1729} & & \\ & & \frac{1}{19} & & & \\ 27 & 3 & & & & \end{array}$$

where $-\frac{6}{1729}$, for instance, is calculated as $(\frac{1}{19} - \frac{1}{7})/(27-1)$.
We can now use equation (2.12) to approximate our test function b

$$y(x) \approx 0 + x.(1) + x(x-1).(-\frac{3}{28}) + x(x-1)(x-8).(\frac{239}{62244}) \tag{3.9}$$

The function is approximated as we need to ignore all further terms, having run out of divided differences. However, simplification of equation (3.9) gives
$y(x) = 0.00383972x^3 - 0.1417004x^2 + 1.1378606x$, which is the standard polynomial (3.3) obtained in section 3.1.1 and already shown to be a poor fit.

3.1.5 Neville's Algorithm

From the definition in section 2.1.5 we form the linear polynomials

$$\begin{array}{rcccl} p_{1,2}(x) & = & \frac{x.1-(x-1).0}{1-0} & = & x \\ p_{2,3}(x) & = & \frac{(x-1).2-(x-8).1}{8-1} & = & \frac{x+6}{7} \\ p_{3,4}(x) & = & \frac{(x-8).3-(x-27).2}{27-8} & = & \frac{x+30}{19} \end{array} \tag{3.10}$$

Using (3.10) we form the quadratic polynomials

$$\begin{array}{rcccl} p_{1,3} & = & \frac{x.p_{2,3}-(x-8).p_{1,2}}{8-0} & = & \frac{-6x^2+62x}{56} \\ p_{2,4} & = & \frac{(x-1).p_{3,4}-(x-27).p_{2,3}}{27-1} & = & \frac{-6x^2+301x+1434}{1729} \end{array} \tag{3.11}$$

Finally, using (3.11) we form the cubic polynomial

$$p_{1,4} = \frac{x.p_{2,4}-(x-27).p_{1,3}}{27-0} = 0.00383972x^3 - 0.1417004x^2 + 1.1378606x \tag{3.12}$$

and hence we have again formed the standard polynomial (3.3).

3.2 Rational Functions

With reference to section 2.2.1 and to the conclusion of section 3.1.1 it seems sensible to test rational forms with the lowest total degree. We will therefore test

$$r_1(x) = \frac{p_1 x + p_0}{x^2 + q_1 x + q_0} \tag{3.13}$$

and

$$r_2(x) = \frac{p_2 x^2 + p_1 x + p_0}{x + q_0} \tag{3.14}$$

Equation (3.13) can be written in the form

$$p_1 x + p_0 = (x^2 + q_1 x + q_0).(r_1(x)) \tag{3.15}$$

and substituting the tabulated points as $r_1(x_i) = y_i$ we get the four linear equations

$$\begin{array}{rcl} 27p_1 + p_0 &=& 3(729 + 27q_1 + q_0) \\ 8p_1 + p_0 &=& 2(64 + 8q_1 + q_0) \\ p_1 + p_0 &=& 1(1 + q_1 + q_0) \\ p_0 &=& 0 \end{array} \tag{3.16}$$

Solving for p_1, p_0, q_1, q_0 gives us

$$r_1(x) = \frac{-157.18182x}{x^2 - 76.36364x - 81.81818} \tag{3.17}$$

and by a similar process we can obtain our interpolating function in the form of (3.14) as

$$r_2(x) = \frac{0.0460251x^2 + 1.8577406x}{x + 0.9037657} \tag{3.18}$$

The graphs of both rational functions and the test function are drawn in figure 3.6.

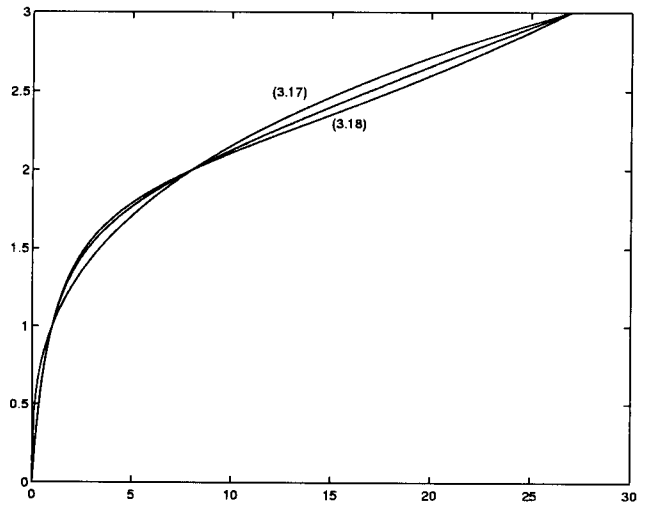

Figure 3.6: Rational interpolating Functions

As can be seen both functions fit quite well, especially (3.18), and they are both much better than any of the polynomial functions tried earlier.

3.2.1 Padé Approximations

As we do not have derivatives for our tabulated data, Padé Approximations will not be compared with our test function.

3.3 Cubic Splines

With four tabulated points we need to find three cubic functions of the form

$$f_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{3.19}$$

across the three intervals. Section 2.3 shows that the values c_i can be found as the solution to the matrix equation (2.25). Substituting the tabulated gives the equation

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 16 & 7 & 0 \\ 0 & 7 & 52 & 19 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} 0 \\ -18/7 \\ -36/133 \\ 0 \end{bmatrix}$$

which has solution $c_1 = 0, c_2 = -0.168352, c_3 = 0.0174574, c_4 = 0$.

We can now substitute into equations (2.22) to find b_i and (2.18) to find d_i, with $a_i = y_i$ to give the three cubic splines

$$\begin{aligned} f_1(x) &= 1.056117x - 0.0561173x^3; \qquad 0 \leq x \leq 1 \\ f_2(x) &= 1 + 0.887765(x-1) - 0.168352(x-1)^2 + 0.00884806(x-1)^3; \qquad 1 \leq x \leq 8 \\ f_3(x) &= 2 - 0.168496(x-8) + 0.0174574(x-8)^2 - 0.000306271(x-8)^3; \qquad 8 \leq x \leq 27 \end{aligned}$$

We can now plot these splines together with our test function for comparison as figure 3.7.

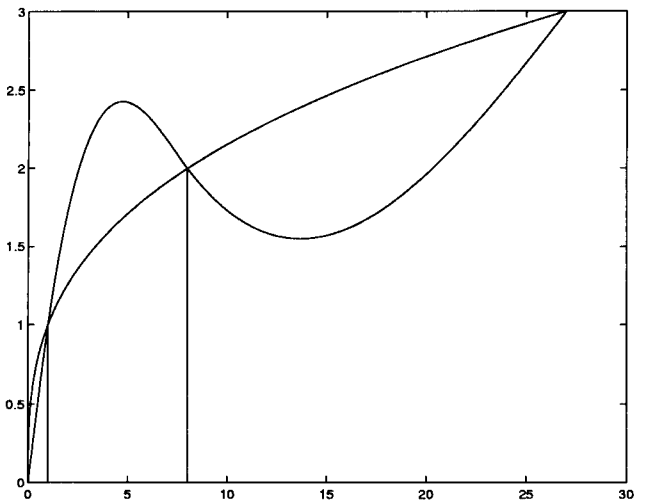

Figure 3.7: Cubic Spline Approximation

Although this gives a better fit than a single polynomial, it is not a particularly good fit.

3.4 Least Squares Methods

Initially we will find a linear approximation of the form $y = ax + b$. To find a and b for the least squares line we need to minimise the expression E where

$$E = \sum_{i=1}^{4} (y_i - ax_i - b)^2 \tag{3.20}$$

$$\frac{\partial E}{\partial b} = 0 \Rightarrow a\sum_{i=1}^{4} x_i + 4b = \sum_{i=1}^{4} y_i \tag{3.21}$$

and

$$\frac{\partial E}{\partial a} = 0 \Rightarrow a\sum_{i=1}^{4} x_i^2 + b\sum_{i=1}^{4} x_i = \sum_{i=1}^{4} x_i y_i \tag{3.22}$$

For our data, $\sum_{i=1}^{4} x_i = 36$, $\sum_{i=1}^{4} y_i = 6$, $\sum_{i=1}^{4} x_i y_i = 98$, $\sum_{i=1}^{4} x_i^2 = 794$ and equations (3.21) and (3.22) become

$$\begin{aligned} 36a + 4b &= 6 \\ 794a + 36b &= 98 \end{aligned}$$

These equations can be solved for a, b giving the least squares line as

$$y = 0.093617x + 0.65745 \tag{3.23}$$

We can now continue by finding the least squares quadratic approximation,

$$y = ax^2 + bx + c \tag{3.24}$$

to our test function. For this we need to find a, b, c to minimise the expression for S given by

$$S = \sum_{i=1}^{4} (y_i - ax_i^2 - bx_i - c) \tag{3.25}$$

Once again we can use calculus, with

$$\frac{\partial E}{\partial c} = 0 \Rightarrow a\sum_{i=1}^{4} x_i^2 + b\sum_{i=1}^{4} x_i + 4c = \sum_{i=1}^{4} y_i \tag{3.26}$$

$$\frac{\partial E}{\partial b} = 0 \Rightarrow a\sum_{i=1}^{4} x_i^3 + b\sum_{i=1}^{4} x_i^2 + c\sum_{i=1}^{4} x_i = \sum_{i=1}^{4} x_i y_i \tag{3.27}$$

$$\frac{\partial E}{\partial a} = 0 \Rightarrow a\sum_{i=1}^{4} x_i^4 + b\sum_{i=1}^{4} x_i^3 + c\sum_{i=1}^{4} x_i^2 = \sum_{i=1}^{4} x_i^2 y_i \tag{3.28}$$

Using our test data equations (3.26), (3.27), (3.28) give the linear equations

$$\begin{aligned} 794a + 36b + 4c &= 6 \\ 20196a + 794b + 36c &= 98 \\ 535538a + 20196b + 794c &= 2316 \end{aligned}$$

These equations can easily be solved to give us the least squares quadratic approximation as

$$y = -0.0062055x^2 + 0.26592x + 0.33852 \tag{3.29}$$

We can now graph our linear and quadratic approximations, again with the test function for comparison.

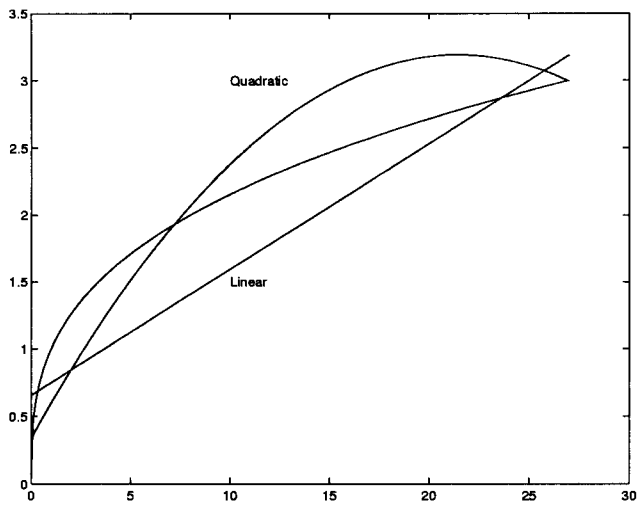

Figure 3.8: Least Squares Approximations

We would not expect the linear fit to be of much use as an approximating function to our curve, but the quadratic is reasonable over the range of values used. However, it can be seen that the quadratic is decreasing at the right end of the domain whereas our test function is increasing.

3.5 Neural Networks

Only the back propagation function approximations can be applied to the test function.

3.5.1 Early Stopping

We will first consider the early stopping routine with the MATLAB function 'traingdx'. The routine will be run with several numbers of neurons in the hidden layer and the results are graphed in figures 3.9 to 3.12.

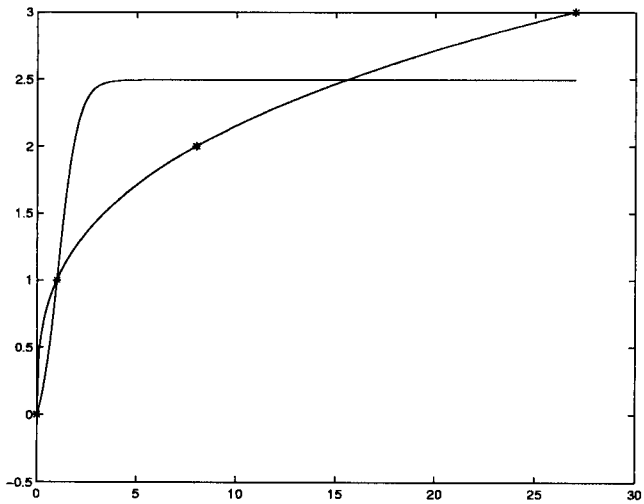

Figure 3.9: 1 hidden neuron

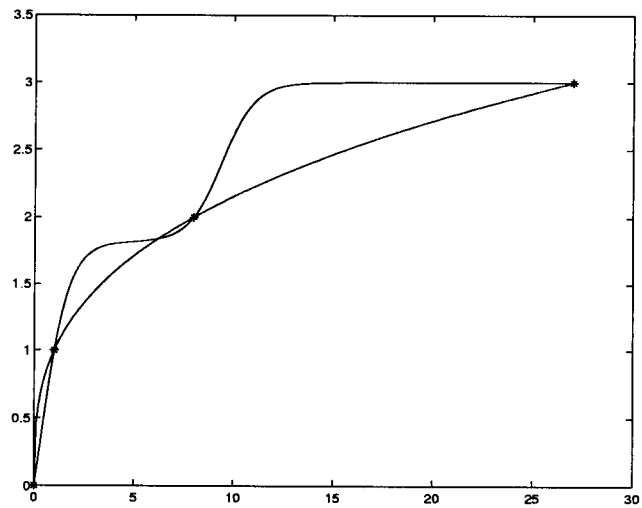

Figure 3.10: 2 hidden neurons

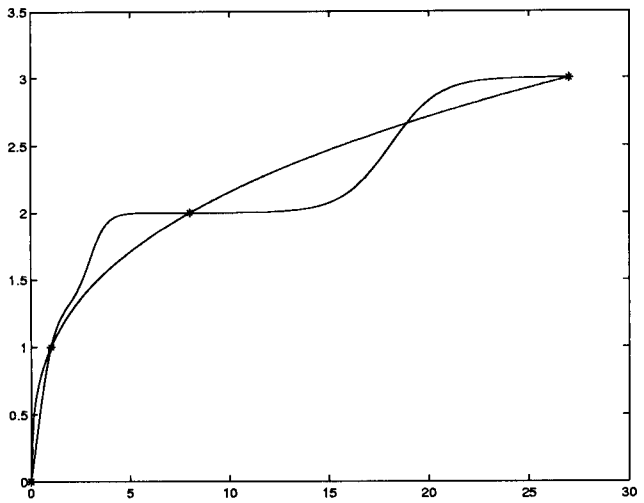

Figure 3.11: 4 hidden neurons

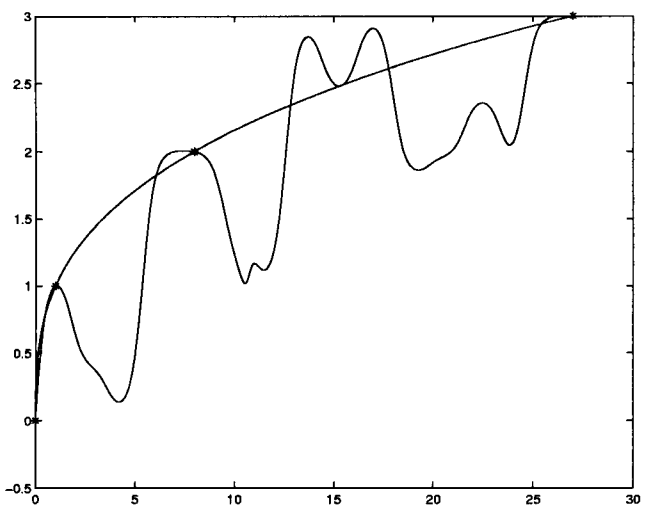

Figure 3.12: 16 hidden neurons

We can see from figure 3.9 that with only 1 neuron we have insufficient degrees of freedom to hit the tabulated points and we have underfitting. Figures 3.10 and 3.11 show a reasonable fit with 2 and 4 neurons respectively, but with 16 neurons we have too many degrees of freedom and figure 3.12 shows clear overfitting.

3.5.2 Generalized Regression

We can see from figure 3.13 that this routine gives a poor approximation to the curve.

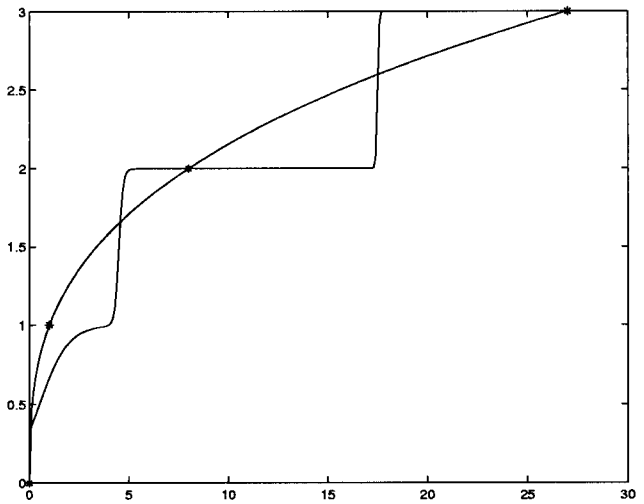

Figure 3.13: GRNN Network Approximation

3.6 Logistic Curve

This curve is for fitting population data and not Mathematical functions and so it would be inappropriate to compare it with our test function. It will be tested on population data in the next chapter.

3.7 Test Conclusions

The various methods have been tested on a smooth, straightforward curve, and the results have been extremely disappointing. Some of the methods, the Hermite especially, have produced large differences at the non tabulated points and even the splines method failed to produce the kind of accuracy required to model populations with any degree of confidence.

The rational functions appear to be the most hopeful for further analysis but some of the other techniques will be applied to real data before they are discarded.

Chapter 4

Testing with Population Data

4.1 Polynomial Approximations

It is sensible to test the polynomial interpolation with real data before dismissing this technique. We will use four sets of data from the table in the appendix to graph the population data together with the polynomial interpolating function.The graph for Warrington, a town close to Chester, is shown below as figure 4.1. This polynomial looks to be quite a good fit to the data.

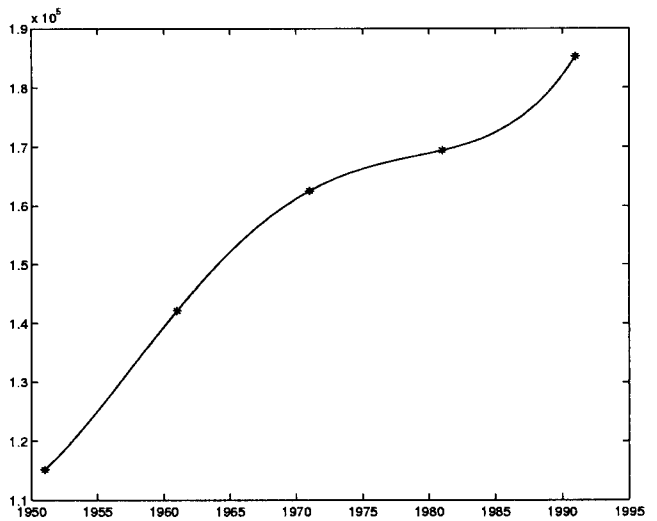

Figure 4.1: Polynomial Approximation for Warrington

Figure 4.2 shows the polynomial approximation to the Exeter data. This fit is not as

successful with an implied fall in population between 1951 and 1961, and again between 1981 and 1991.

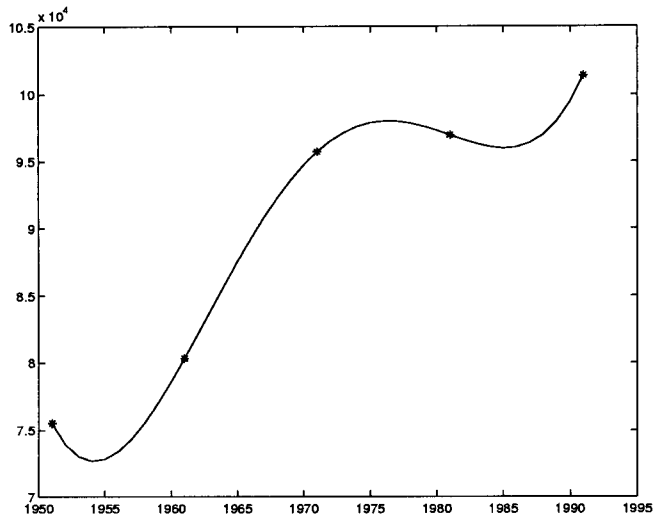

Figure 4.2: Polynomial Approximation for Exeter

A similar picture, but with more substantial falls can be seen in the graph for the Bath approximation as shown in figure 4.3.

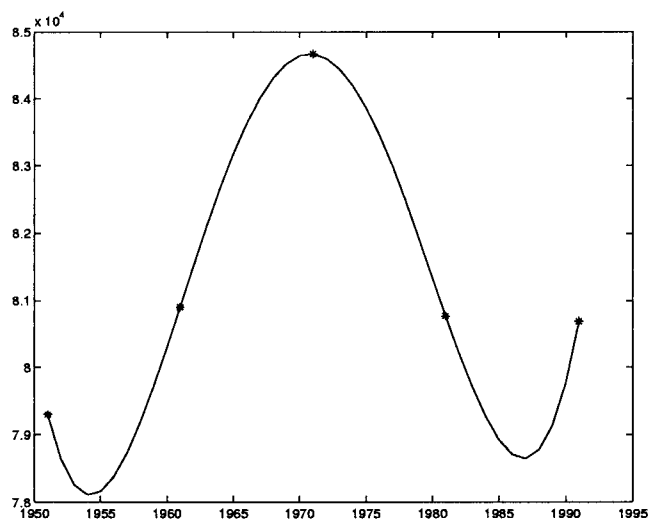

Figure 4.3: Polynomial Approximation for Bath

Finally we have a graph which shows a reasonable fit to the data of Chester in figure 4.4. Also the population levels between 1981 and 1991 actually show a decrease as indicated

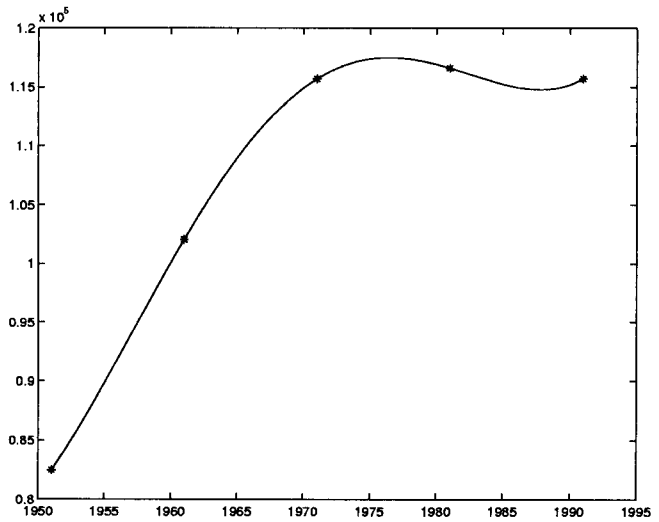

Figure 4.4: Polynomial Approximation for Chester

by the intercensal estimates produced by the government. It will therefore be of interest to compare this period in more detail.

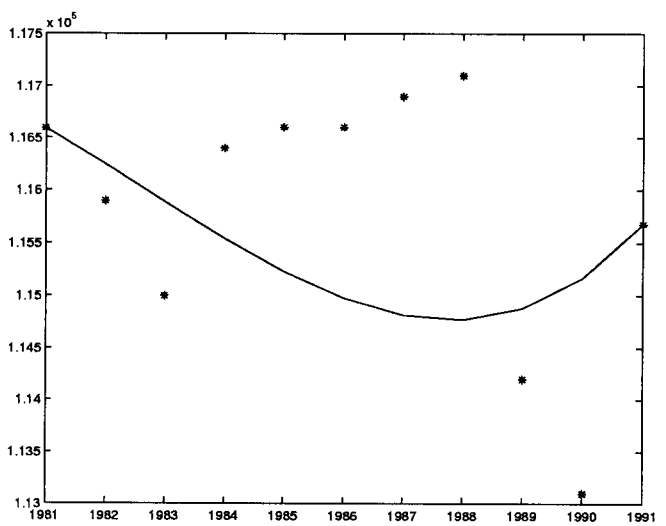

Figure 4.5: Detailed Polynomial Approximation for Chester

However, we can see from the more detailed graph, figure 4.5, that the fit between the interpolating polynomial and the government estimates for the period 1981 to 1991 is actually very poor.

The fit of the interpolating polynomial was generally much better than the work on the

test function suggested. It is far from satisfactory, but it will be considered later in this project when extrapolation is required.

4.2 Rational Approximations

We will test three sets of data with the three rational forms

$$y_1(x) = \frac{ax^3 + bx^2 + cx + d}{x + e} \tag{4.1}$$

$$y_2(x) = \frac{ax^2 + bx + c}{x^2 + dx + e} \tag{4.2}$$

$$y_3(x) = \frac{ax + b}{x^3 + cx^2 + dx + e} \tag{4.3}$$

We once again find the five coefficients in the rational form by substituting the five data points and solving five linear equations. The plots for Warrington are shown as figures 4.6 to 4.8.

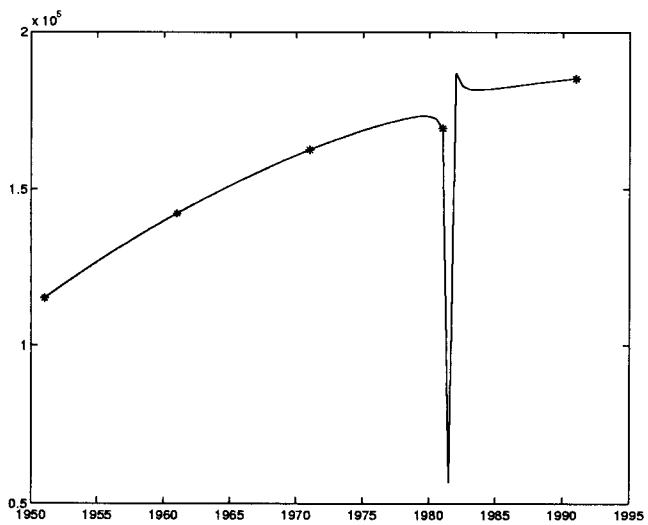

Figure 4.6: Warrington y_1

Both y_1 and y_2 have discontinuities as can be seen in figures 4.6 and 4.7 and so are unsuitable as interpolating functions.

Function y_3, however, as figure 4.8 indicates, is a good fit and will be considered in the extrapolation chapter.

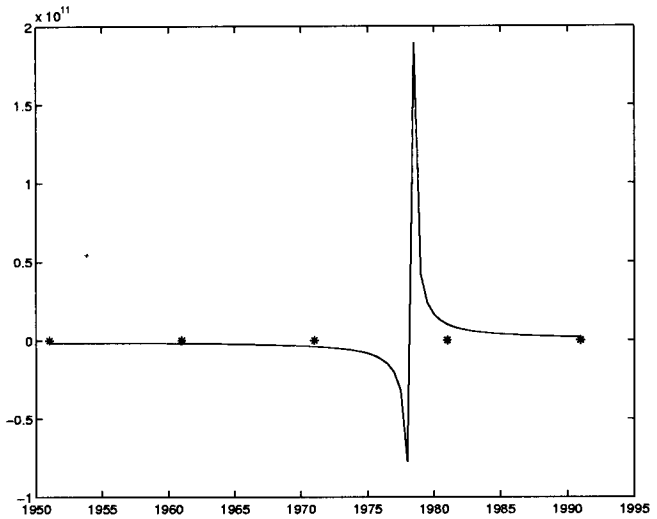

Figure 4.7: Warrington y_2

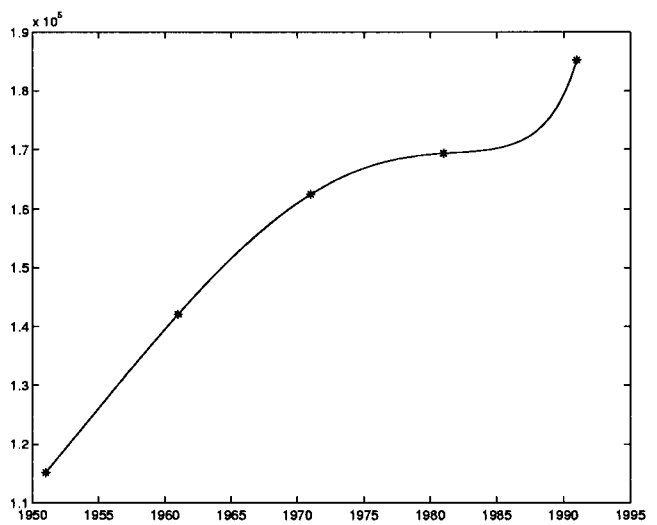

Figure 4.8: Warrington y_3

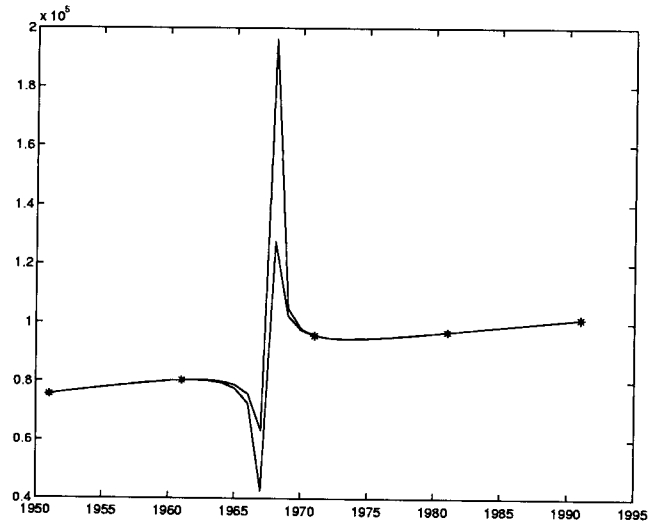

Figure 4.9: Exeter y_1, y_2

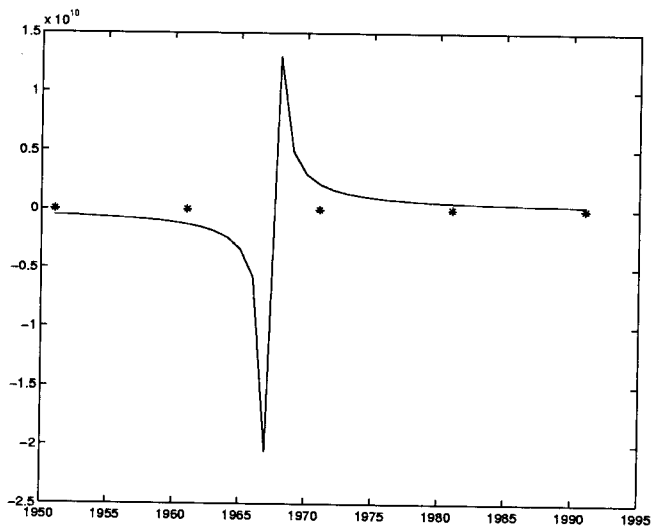

Figure 4.10: Exeter y_3

All three forms for the Exeter data shown in figures 4.9 and 4.10 contain discontinuities, and the linear equations to find y_3 are so ill-conditioned that the function calculated fails to pass through the data points.

We conclude this section by considering the Chester data. We can see from figure 4.11 that y_1 has a discontinuity and from figure 4.12 that y_2 is very ill-conditioned. The final form, figure 4.13, fits well and it will also be considered during extrapolation.

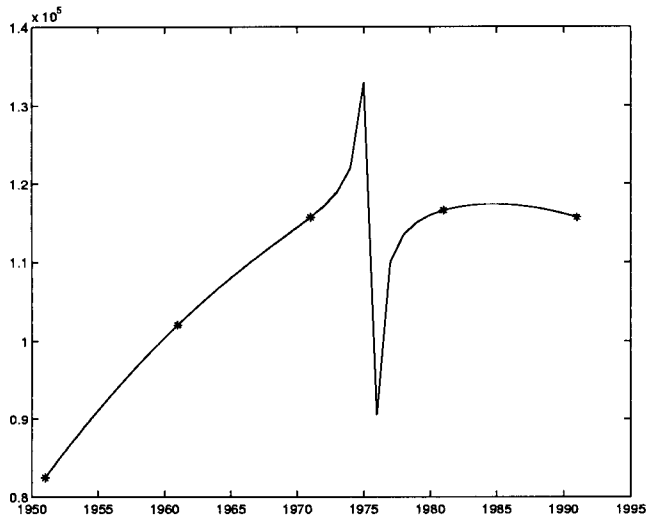

Figure 4.11: Chester y_1

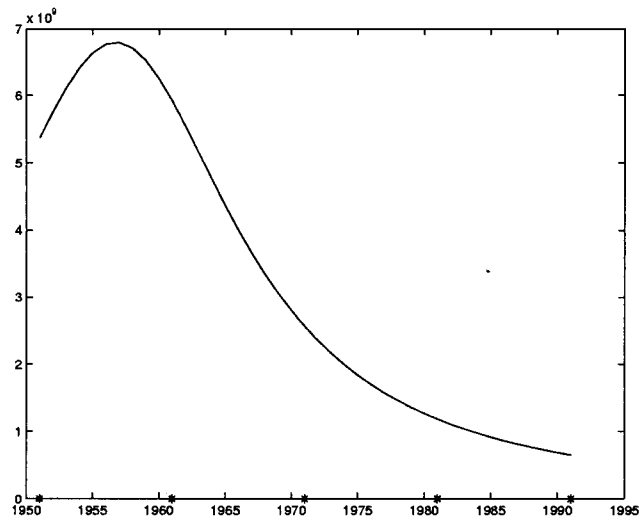

Figure 4.12: Chester y_2

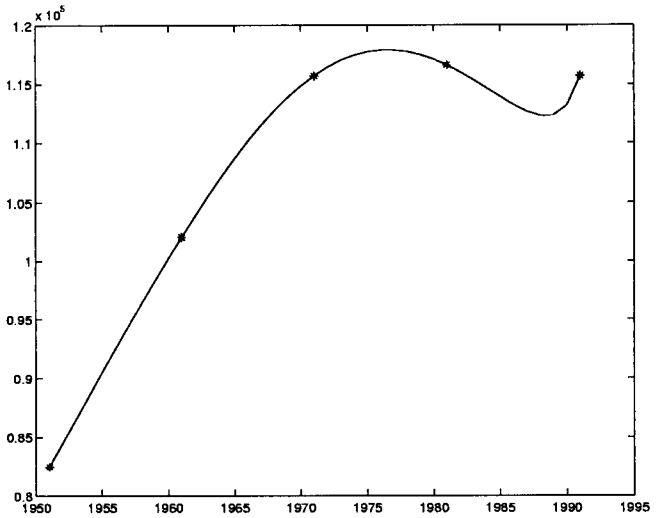

Figure 4.13: Chester y_3

However, the analysis on the three sets of data indicates that it is highly unlikely that rational approximations will produce a satisfactory population model.

Chapter 5

Neural Networks

All the routines used in this chapter are discussed in detail in the Neural Network Toolbox [2].

All the MATLAB programs for this chapter are included in the appendix. To simplify the programming the census years 1951-1991 have been entered as 0 - 4 and the population figures have been entered in 100000's.

All three neural networks described earlier will be tested in this chapter. Each has been demonstrated with several different numbers of hidden neurons. This was implemented to observe the effect of varying this number and also to see if an optimum number could be selected.

5.1 Back Propogation Function Approximation

In this section we will look at two routines for function approximation.

5.1.1 Automated Regularization

The Bayesian regularization routine has been implemented by the MATLAB function 'trainbr' to train the network. The program was then run with 4, 20 and finally 40 inner neurons for the population of Bournemouth given in the appendix. In each case the graph showed an underfitting of the data. Two of these graphs are included below as figures 5.1 to 5.2.

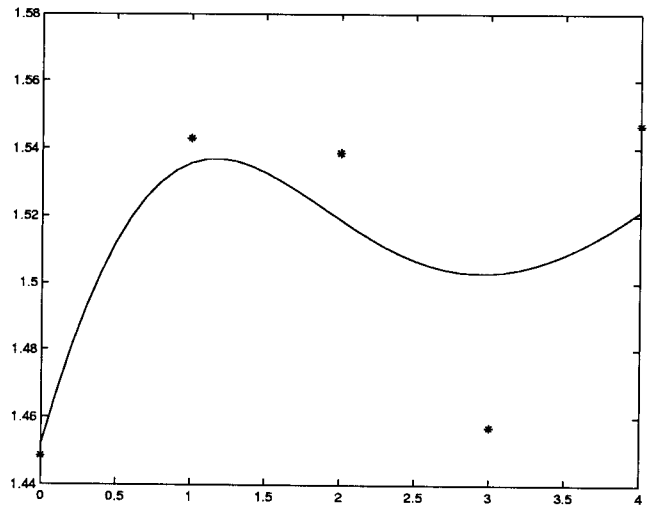

Figure 5.1: 4 hidden neurons

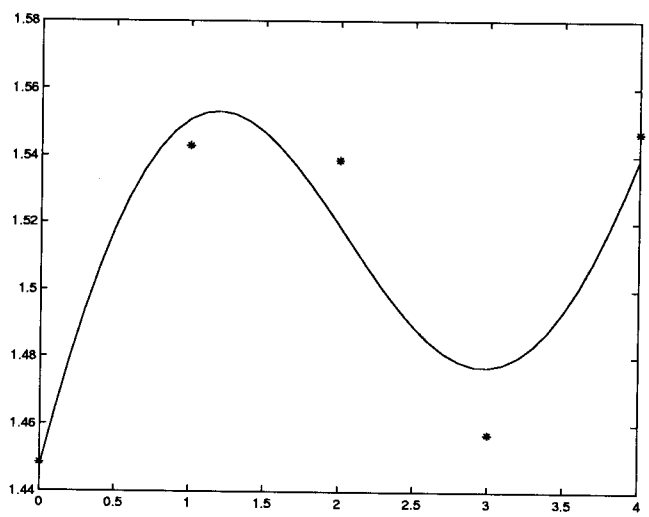

Figure 5.2: 20 hidden neurons

The fit can be seen to be inadequate for any useful population predictions.

5.1.2 Early Stopping

This routine can be used with any of the training functions used by MATLAB, and the function used below was 'traingdx'. The data for Bournemouth was again used to check the fit of this method, shown in figures 5.3 to 5.5.

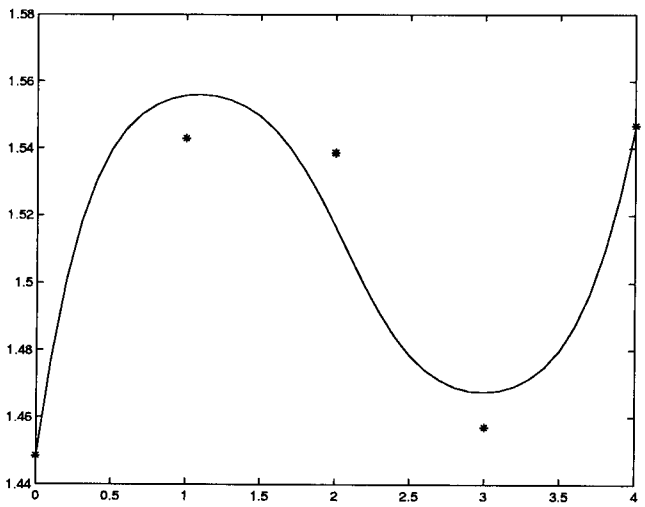

Figure 5.3: 4 hidden neurons

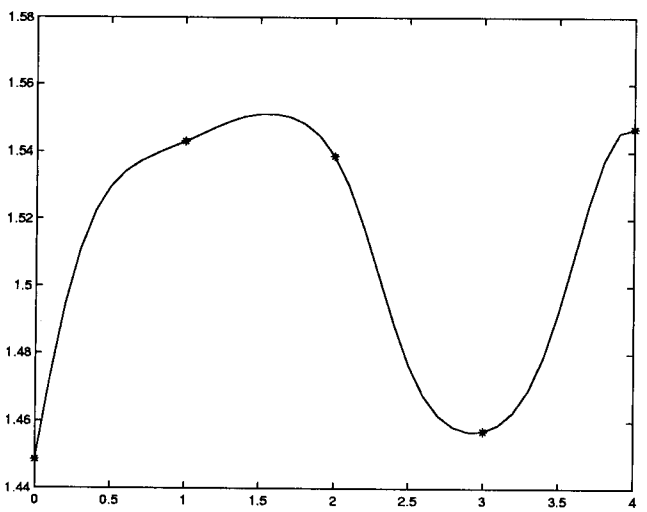

Figure 5.4: 6 hidden neurons

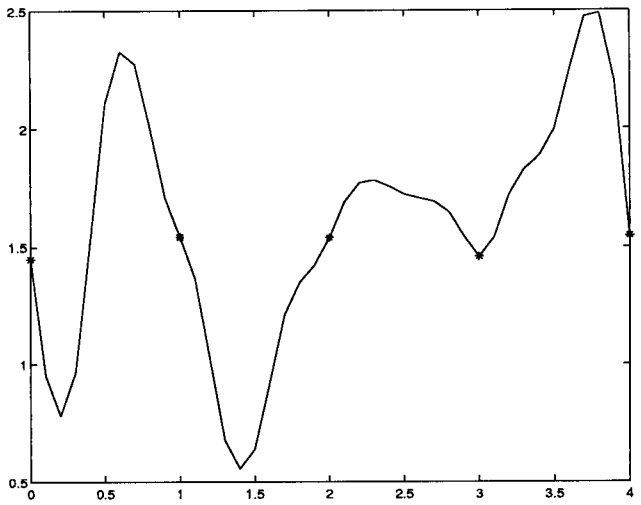

Figure 5.5: 16 hidden neurons

The results show an underfitting with 4 hidden neurons, indicating a larger hidden layer is required. The first fit occurs with a hidden layer of 6 neurons and once this layer is increased to 16 overfitting becomes the problem, as is demonstated by the figure 5.5. Only the 6 neuron figure indicates a decent fit to the data, and this approximation will be considered at the extrapolation stage of this project.

It is evident from the figures contained in this section that the use of back propagation function approximation neural networks to approximate population data is unlikely to succeed. There is either underfitting with errors of up 5000 persons, or overfitting as shown in figure 5.4 which predicts unacceptable intercensal figures.

5.2 Generalized Regression

A generalized regression network was created using the MATLAB function 'newgrnn' and this approximating function was plotted with the Bournemouth data as figure 5.6.

The approximating curve fits smoothly between the census data points, but the difference between the data and the approximating values is far too large for use in interpolating any population estimates.

5.3 Back Propagation Vectors

This technique looks for patterns in a block of the data given in Appendix B from some of the areas and some or all of the census years 1951 -1981. It trains these figures onto

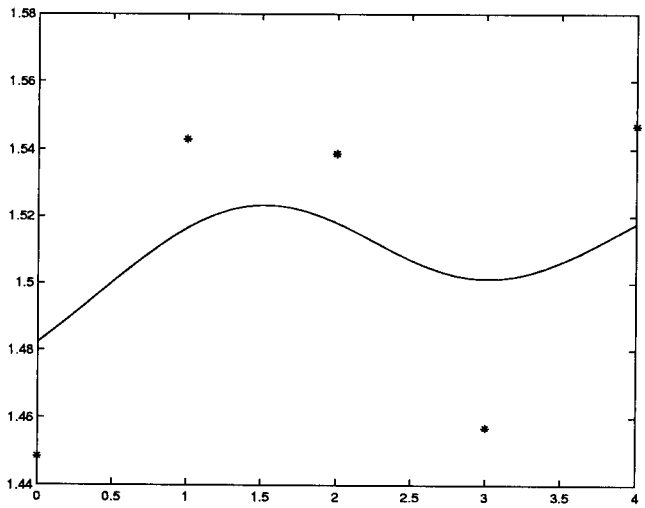

Figure 5.6: GRNN Network approximation

the targets of the census data for 1991. We can then input a similar size block from different areas for 1951 - 1981 and the network will predict the 1991 population figures. This technique is moving us into the final phase of this project as we are now extrapolating data. However, this is not the type of extrapolating that we require, as to use this method to estimate 2001 data, and the data for years from 1991 to 2001, we need to have population data for 2001 for our set of training areas. This is clearly not a feasible requirement.

5.3.1 Gradient Descent with Momentum

We can use the method of gradient descent with momentum to train our input vectors. The method was applied to various sizes and numbers of input vectors. The accuracy of the training and the accuracy of the predictions can be checked in the following sets of tables. The number of neurons in the hidden layers have also been varied for further comparisons. The program also produces a graph to show the error bounds at each training pass (epoch), and some of these graphs are included for information.

4 by R^2 input

Initially the data for 4 areas, Eastbourne, Exeter, Isle of Wight and Leicester was used from 1971 and 1981 to train onto the 1991 data. Once trained, the network was fed the input data from Chester, Bath, Bournemouth and Cheltenham for 1971 and 1981 in order to predict the 1991 levels. The training was completed over 1000 passes (epochs). The following tables, 5.1 to 5.8, show the output for 3, 4, 20 and 40 hidden neurons.

	Eastbourne	Exeter	Isle of Wight	Leicester
Target	78778	101395	127232	272133
Output	76398	107323	123676	272104
Error	-2380	5928	-3556	-29

Table 5.1: Training for 3 hidden layers

	Chester	Bath	Bournemouth	Cheltenham
Actual	115680	80689	154677	103566
Predicted	129185	93262	170632	96610
Error	13505	12573	15955	-6956

Table 5.2: Predicting for 3 hidden layers

	Eastbourne	Exeter	Isle of Wight	Leicester
Target	78778	101395	127232	272133
Output	79094	101603	126720	272190
Error	286	208	-512	57

Table 5.3: Training for 4 hidden layers

	Chester	Bath	Bournemouth	Cheltenham
Actual	115680	80689	154677	103566
Predicted	131147	91114	213237	91940
Error	15467	10425	58560	-11626

Table 5.4: Predicting for 4 hidden layers

Note that we have nearly hit the target with four neurons, but the predicted values are very inaccurate.

Further running of the program showed that as the number of hidden neurons was

	Eastbourne	Exeter	Isle of Wight	Leicester
Target	78778	101395	127232	272133
Output	78433	101896	127063	272129
Error	-345	501	-169	-4

Table 5.5: Training for 20 hidden layers

	Chester	Bath	Bournemouth	Cheltenham
Actual	115680	80689	154677	103566
Predicted	125162	89818	145540	102999
Error	9482	9129	-9137	-567

Table 5.6: Predicting for 20 hidden layers

	Eastbourne	Exeter	Isle of Wight	Leicester
Target	78778	101395	127232	272133
Output	78772	101395	127232	272133
Error	-6	0	0	0

Table 5.7: Training for 40 hidden layers

	Chester	Bath	Bournemouth	Cheltenham
Actual	115680	80689	154677	103566
Predicted	111582	71630	271222	113178
Error	-4098	-9059	116545	9612

Table 5.8: Predicting for 40 hidden layers

increased the network trained onto the targets more accurately. However the predictions of the 1991 data was significantly different from the actual values.

4 by R^4 input

We now repeat this investigation with the same areas, but using increased data for training and predicting. The four census values for 1951,1961,1971 and 1981 will be used as input vectors, and again with a selection of hidden layer sizes, containing 5, 7, 25 and 50 neurons. Tables 5.9 to 5.16 display the results.

	Eastbourne	Exeter	Isle of Wight	Leicester
Target	78778	101395	127232	272133
Output	80558	97986	128937	272059
Error	1780	-3409	1705	-74

Table 5.9: Training for 5 hidden layers

	Chester	Bath	Bournemouth	Cheltenham
Actual	115680	80689	154677	103566
Predicted	115774	102924	186211	100131
Error	94	22235	31534	-3435

Table 5.10: Predicting for 5 hidden layers

	Eastbourne	Exeter	Isle of Wight	Leicester
Target	78778	101395	127232	272133
Output	78768	101411	127226	272132
Error	-10	16	-6	-1

Table 5.11: Training for 7 hidden layers

For the training shown in table 5.11, resulting in predictions given in table 5.12, we can include a graph, figure 5.7, which demonstrates the reduction of error during each epoch of the training routine. The error goal was reached after 503 passes.

We can see that in all four cases the predictions contain errors in at least one area of 30000 or more. The 5 neuron case predicted the Chester population to an error of less than

	Chester	Bath	Bournemouth	Cheltenham
Actual	115680	80689	154677	103566
Predicted	128508	93356	246310	96011
Error	12828	12667	91633	-7555

Table 5.12: Predicting for 7 hidden layers

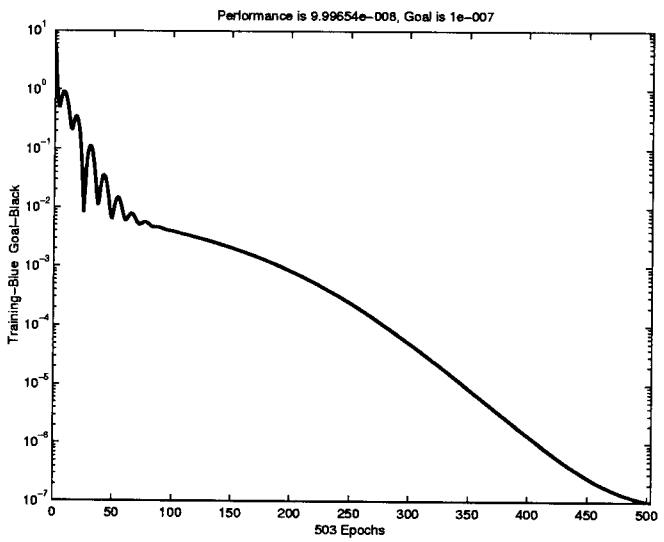

Figure 5.7: Error at each pass with 7 hidden neurons.

	Eastbourne	Exeter	Isle of Wight	Leicester
Target	78778	101395	127232	272133
Output	78646	101695	127056	272137
Error	-132	300	-176	4

Table 5.13: Training for 25 hidden layers

100. However, the levels for the other three populations show large errors which implies that the accurate Chester result was a fluke.

	Chester	Bath	Bournemouth	Cheltenham
Actual	115680	80689	154677	103566
Predicted	116071	85463	204778	98100
Error	391	4774	50101	-5466

Table 5.14: Predicting for 25 hidden layers

	Eastbourne	Exeter	Isle of Wight	Leicester
Target	78778	101395	127232	272133
Output	78750	101446	127209	272133
Error	-28	51	-23	0

Table 5.15: Training for 50 hidden layers

	Chester	Bath	Bournemouth	Cheltenham
Actual	115680	80689	154677	103566
Predicted	133647	77671	44781	100941
Error	17967	-3018	-109896	-2625

Table 5.16: Predicting for 50 hidden layers

6 by R^4 input

To increase the input data further the number of areas was increased from four to six. We used 8 hidden neurons for this run. The program was run over a smaller stopping tolerance and the routine took 31800 epochs and several hours to reach this goal. The results in tables 5.17 and 5.18 show a very poor prediction of the actual values even though the training was very accurate.

5.3.2 Levenberg-Marquardt

As the previous routine was becoming very slow, a new training routine was tried. The MATLAB function 'trainlm', using the Levenberg-Marquardt algorithm was chosen and run with 2, 10, 15 and 20 hidden neurons. This routine reached an even smaller stopping

	Eastbourne	Exeter	Isle of Wight	Leicester	Norwich	Oxford
Target	78778	101395	127232	272133	122661	124058
Output	78775	101402	127418	272133	122659	124074
Error	-3	7	186	0	-2	16

Table 5.17: Training for 6 by R^4 input

	Chester	Bath	Bournemouth	Cheltenham	Crewe/Nantwich	York
Actual	115680	80689	154677	103566	105400	101436
Predicted	132356	88812	218400	90737	102824	116415
Error	16676	8123	63723	-12829	-2576	14979

Table 5.18: Predicting for 6 by R^4 input

tolerance in a maximum of just 6 epochs. The output hit the training target to the nearest person on all four occasions. The slowest case was with 10 neurons, and the error graph is shown as figure 5.8.

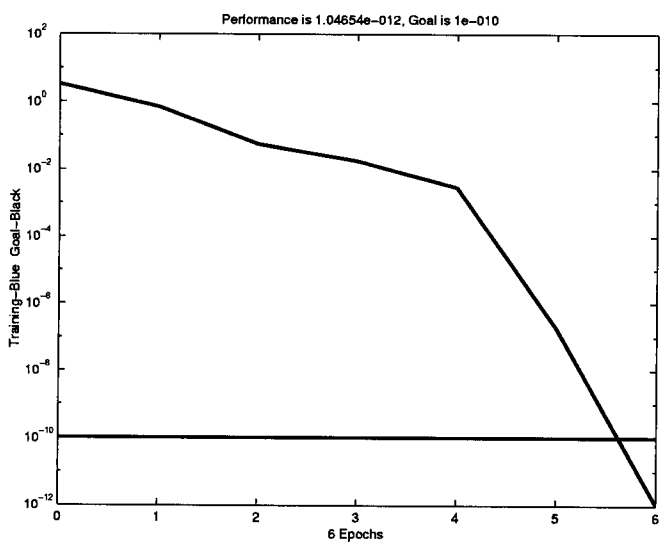

Figure 5.8: Error at each pass with 10 hidden neurons.

With the target being hit each time there is only the need to look at the prediction

errors for this routine. We can see that once again the predictions differ from the actual populations by a considerable margin.

	Chester	Bath	Bournemouth	Cheltenham	Crewe/Nantwich	York
Actual	115680	80689	154677	103566	105400	101436
Predicted	116628	91441	151832	97074	102774	122578
Error	948	10752	-2845	-6492	-2626	21142

Table 5.19: Predicting by Levenberg-Marquardt with 2 neurons

	Chester	Bath	Bournemouth	Cheltenham	Crewe/Nantwich	York
Actual	115680	80689	154677	103566	105400	101436
Predicted	117062	96757	157963	99103	106106	116672
Error	1382	16068	3286	-4463	706	15236

Table 5.20: Predicting by Levenberg-Marquardt with 10 neurons

	Chester	Bath	Bournemouth	Cheltenham	Crewe/Nantwich	York
Actual	115680	80689	154677	103566	105400	101436
Predicted	111285	93368	197354	95624	101694	128626
Error	-4395	12679	42677	-7942	-3706	27190

Table 5.21: Predicting by Levenberg-Marquardt with 15 neurons

	Chester	Bath	Bournemouth	Cheltenham	Crewe/Nantwich	York
Actual	115680	80689	154677	103566	105400	101436
Predicted	104071	98766	202074	95416	101678	130002
Error	-11609	18077	47397	-8150	-3724	28566

Table 5.22: Predicting by Levenberg-Marquardt with 20 neurons

Chapter 6

The Logistic Curve

In this chapter we will attempt to fit logistic curves to some of the data from the appendix. If we consider the method of section 2.6 for the data for Warrington, for instance, we need to plot the graph of P_{i+1}/P_i against P_{i+1}, which will produce a straight line if the model is appropriate. The table below shows the data converted for this approach.

	1951	1961	1971	1981
P_{i+1}/P_i	1.2340	1.1435	1.0422	1.0935
P_{i+1}	142113	162507	169372	185200

We can now plot this data as figure 6.1. We can include on this graph the least squares regression line in order to check for linearity. This is $y = -0.0000037x + 1.7375$

We can clearly see that the fit is not good. As a further check we can calculate Pearson's correlation coefficient, which gives $r = -0.8081$, and for 4 data values there is insufficient evidence to reject the null hypothesis of $r = 0$. For a two tail test the critical value for r is 0.950. Consequently the Warrington data does not appear to fit a logistic model.

It is appropriate to repeat this analysis with three further sets of data to confirm that in general census population data does not follow the logistic model. We will only include the graphs, figures 6.2 to 6.4, and the regression coefficient with each analysis.

We have r values of -0.0188, 0.7216 and -0.8466 respectively, and in all three cases the evidence is strongly against a logistic model. There is clearly no point in following this model any further.

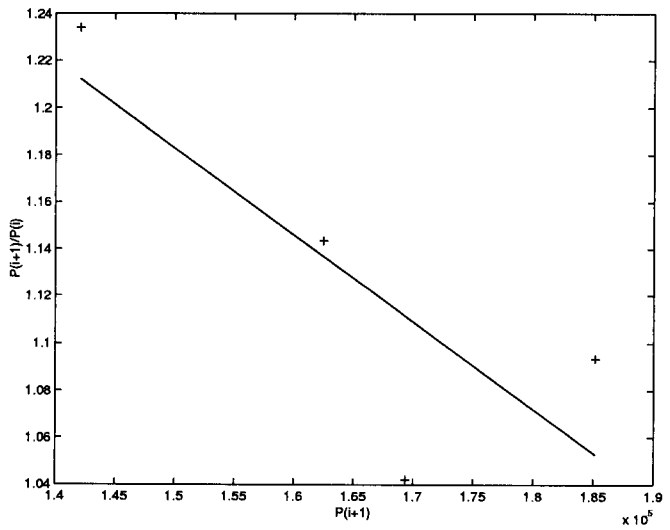

Figure 6.1: Logistic check for Warrington

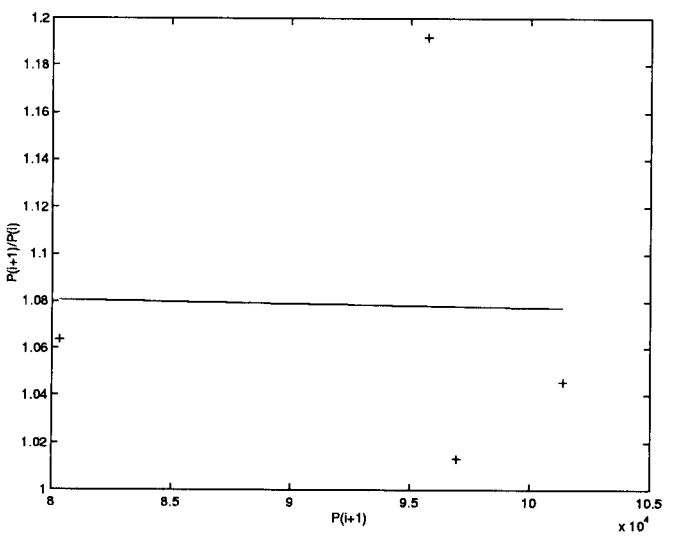

Figure 6.2: Logistic check for Exeter

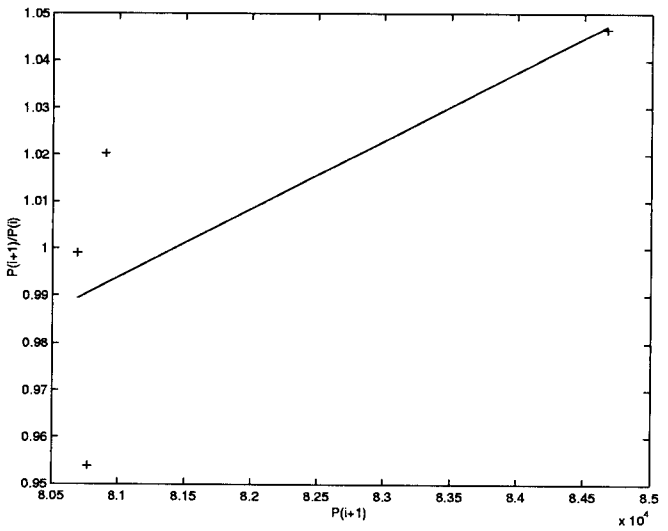

Figure 6.3: Logistic check for Bath

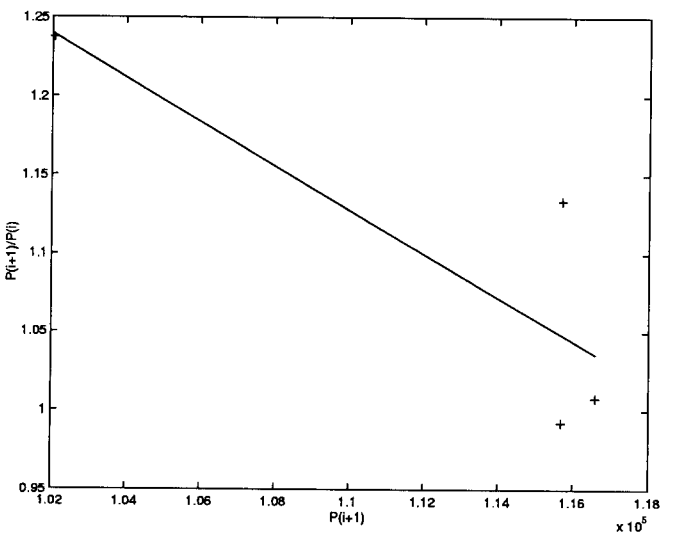

Figure 6.4: Logistic check for Chester

Chapter 7

Extrapolation

In this chapter we will apply the more successful interpolation techniques to the data for Chester to extrapolate the 2001 population levels. We will plot the graphs for each method and finally summarise with a table to compare the various estimates.

7.1 Polynomial Approximations

The graph, figure 7.1, which is an extension of figure 4.4, shows that the term in x^4 is starting to dominate and the curve is unrealistic. It gives a population of 141757 for the year 2001, an increase of over 22 per cent from 1991.

7.2 Rational Approximations

Only the y_3 type of rational form shown in figure 4.13 gave us a reasonable interpolation. The function was calculated as

$$y_3(x) \approx \frac{1.664x - 3318.529}{10^{-8}x^3 - 0.00005944x^2 + 0.1178x - 77.826} \tag{7.1}$$

However, the function $y_3(x)$ in figure 7.2, which resulted from the solution of a set of ill-conditioned linear equations, shows a discontinuity and the gives us an unacceptable method for predicting the 2001 level.

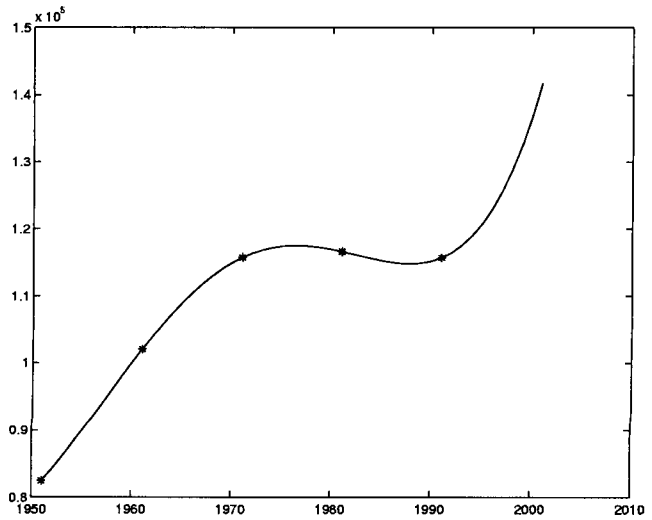

Figure 7.1: Polynomial Extrapolation for Chester

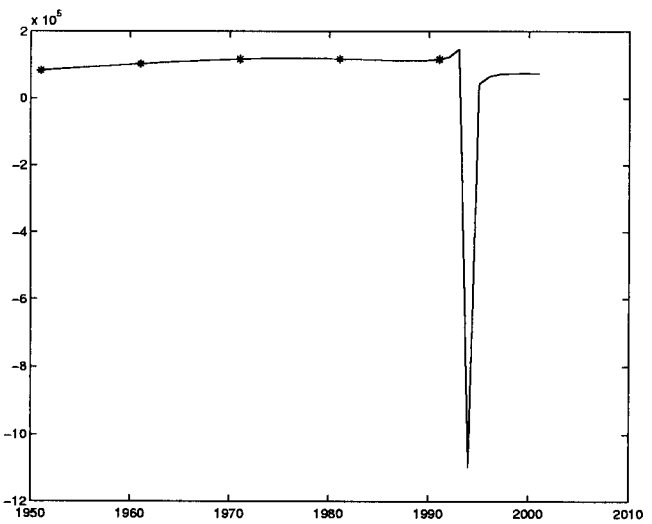

Figure 7.2: Rational Approximation for Chester, $y_3(x)$.

7.3 Neural Networks

7.3.1 Early Stopping

The early stopping routine with 'traingdx' was used with 4, 5 and 6 hidden neurons, in line with the implications of the results from the application of the routine to the Bournemouth

data in subsection 5.1.2. The graphs are shown in figures 7.3 to 7.5.

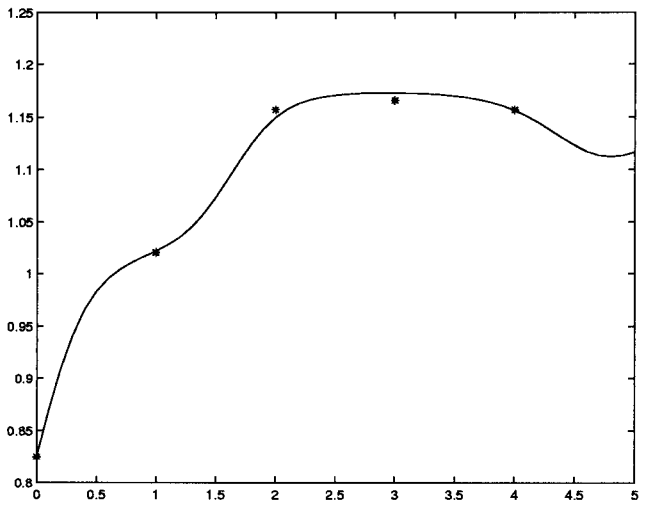

Figure 7.3: Early Stopping for Chester with 4 Neurons

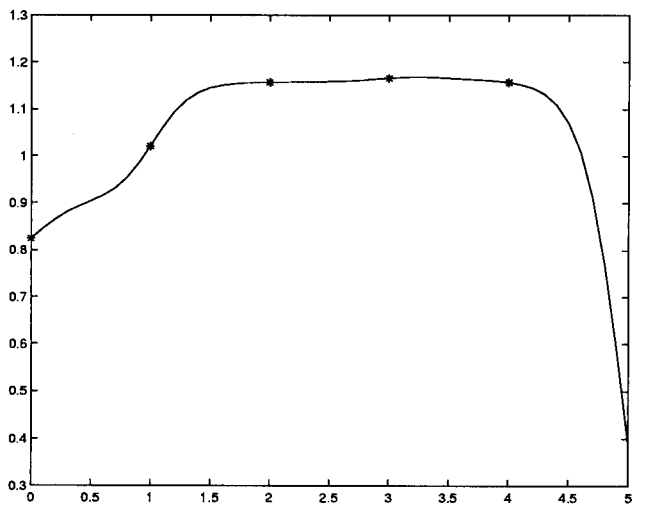

Figure 7.4: Early Stopping for Chester with 5 Neurons

The four neuron routine underfits the data although it does give a feasible prediction of 111696. The two routines that fit the data predict population levels of 38236 and -57188. These results question the validity of any neural network predictions using the supplied data.

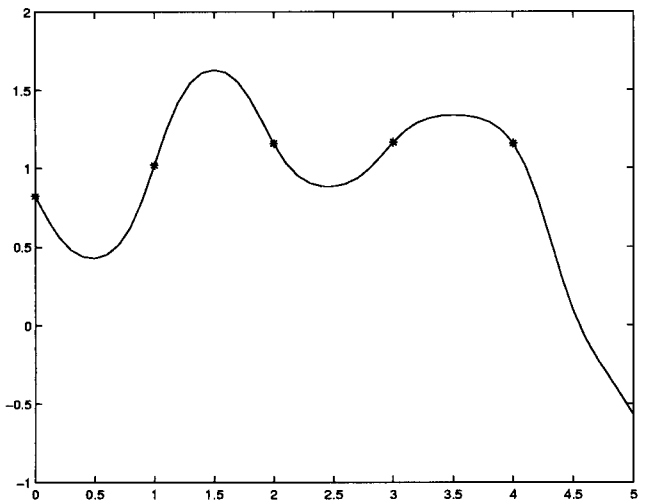

Figure 7.5: Early Stopping for Chester with 6 Neurons

7.3.2 Generalized Regression

The result of this routine is shown in figure 7.6. This routine does not produce a graph passing through the tabulated points. However, it does appear to follow a more acceptable shape for the current data. The predicted value for 2001 is 115781, an increase of just over 100 from the previous census date.

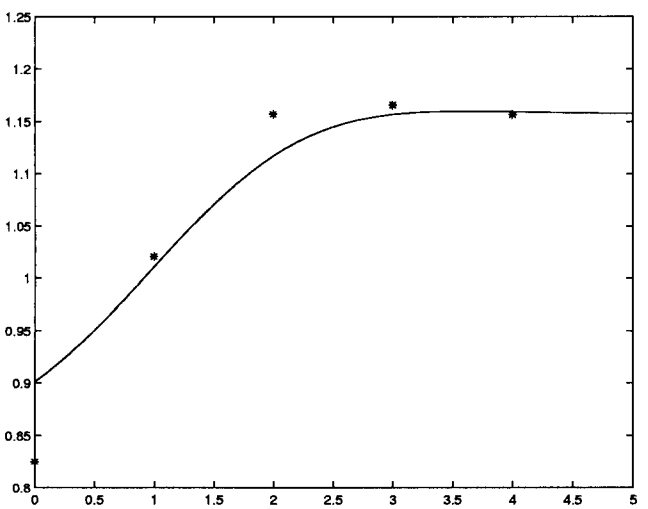

Figure 7.6: GRNN Routine for Chester.

7.4 Summary Table

The table of the predicted population of Chester for 2001 demonstrates a complete lack of consistency across the methods.

Method	Predicted Population
Polynomial Approximation	141757
Rational Approximation	n/a
Neural Network, 4 neurons	111696
Neural Network, 5 neurons	38236
Neural Network, 6 neurons	-57188
Neural Network, GRNN	115781

Chapter 8

Conclusion

We can summarise the results of applying all the techniques described in chapter 2

8.1 Polynomial Methods

8.1.1 General Polynomials

These are easy to formulate by a number of methods but the functions tend to oscillate. At large values of x the term in the highest degree will start to dominate, which makes this technique inappropriate for accurate extrapolation, as figure 7.1 shows.

8.1.2 Rational Functions

Although these functions do not oscillate, they can contain discontinuities. Every rational function used in this project did in fact contain discontinuities and hence this made their use inappropriate.

8.2 Cubic Splines

These are also liable to oscillations and our fit of the test function, figure 3.7, shows this tendency. The final spline could be used for extrapolation but this presents the same dominance problem as described in section 8.1.1.

8.3 Least Squares

The trends in data seem to vary from area to area, and the degree of the regression function, linear, quadratic or higher, will also vary. Least squares approximations will miss many of the tabulated points and will induce errors here and at adjacent points.

8.4 Neural Networks

These have been examined and tested in detail. The function approximations are too varied to be used with any confidence and the vector methods are far too inaccurate.

8.5 Logistic Curve

The graphs in chapter 6 showed clearly that the population data does not follow a logistic model.

8.6 Summary

None of the methods analysed to predict the 2001 population level have produced satisfactory results. The data shows some patterns, but as the neural network routines indicate, there is too much randomness for mathematical methods to succeed. There does not appear to be a cheap and easy way of predicting mathematically intercensal data consistently across all the census areas.

Consequently it is recommended that Chester City Council check the government's methodology and data collection if they wish to dispute the estimates of Chester's intercensal population levels. The methodology appears to be working, with an average error of only 2 per cent over the latest intercensal period to be analysed. The government's formula described in the introduction should accurately incorporate the deaths and births in the period and the National Health Service Central Register should give acceptable estimates of the other variables. However, the errors of up to 12 per cent admitted by the government for some regions do give cause for concern.

It does not appear that the formula is incorrect and the most likely cause of error will be in the actual collection of the data. The next step in Chester City Council's investigation will be to verify the statistics obtained by the government to increment the successive estimates. This could well be a time consuming and expensive operation and the cost could exceed the gains.

Appendix A

Standard Spending Assessment

The following abstract is taken from the Standard Assessment Handbook 1997/98 [7]

Standard Spending Assessments (SSAs) are used to distribute Revenue Support Grant (RSG) to local authorities. An SSA is the Government's assessment of the appropriate amount of revenue expenditure which would allow the authority to provide a standard level of service, consistent with the Government's view of the appropriate amount of revenue expenditure for all local authorities. The calculation of an authority's SSA follows general principles applied equally to all authorities and takes account of each authority's demographic, geographic, social and economic characteristics. Differences in SSAs between authorities with the same service responsibilities are thus due solely to differences in their underlying characteristics.

Revenue Support Grant (RSG) is distributed so that if all local authorities were to spend at the level of their SSA then broadly the same level of council tax could be set in all areas for dwellings in the same valuation band. Consequently RSG equalises for the differences in assessed costs between areas (the SSAs) leaving council tax payers everywhere to pay broadly the same council tax for their valuation band and receive the same standard of service. The RSG entitlement for an area is given by:

RSG = SSA - Income from non-domestic rates - Income from Council Tax for Standard Spending

The Income from Non-domestic Rates (NDR) and Income from Council Tax for Standard Spending (CTSS) are set nationally at about £245.93 per head and £593.09 per band D dwelling, respectively. However, in England local services are provided by more than one tier of local authority. For example, in Northumberland police services are provided by Northumbria Police Authority whilst education, social services, highways and fire are the responsibility of Northumberland County Council, with the district councils being responsible for services such as planning, environmental health, refuse collection and parking. Consequently both the £245.93 per head for non-domestic rates and the council tax of £593.09 per band D dwelling for spending at SSA are divided between these tiers of authorities on

the basis of SSA service shares.[1]

The SSA Handbook shows standard spending assessments for each authority and includes analyses of why SSAs per head differ between authorities. It also provides a summary of any changes in methodology since the 1996/97 Handbook was published. Each authority has been provided with the underlying indicator data necessary to derive its own SSA components from the formulae in the Local Government Finance Report. Data for all authorities can be obtained from Standard Spending Indicator 1997/98 to be published by the Society of County Treasurers.

The calculation of SSAs. *The detailed basis for calculating SSAs is set out in The Local Government Finance Report England) 1997/98. Additional information on the derivation of the formulae for the SSA elements is in the DoE booklet Standard Spending Assessments: Guide to Methodology (1996/97 edition, 1997/98 edition to be published). A brief guide to the calculation of SSAs is given in Appendix II to this handbook (page 449).*

SSAs are built up from separate elements for the following major service blocks and sub-blocks:

1. Education
(i) primary
(ii) secondary
(iii) post-16
(iv) under 5
v) other
2. Personal Social Services
(i) children
(ii) elderly
(iii) other
3. Police
4. Fire
5. Highway Maintenance
6. All Other Services (a collection of smaller services)
7. Capital Financing

The sum of all the SSA elements produces a single SSA for each authority which is then used as a basis for distributing Revenue Support Grant. Local authorities, however, retain discretion over their expenditure priorities both between and within services.

The sum of SSA elements across all services for all authorities equals Total Standard Spending (TSS) net of specific and special grants. TSS represents the Government's view of the appropriate amount of spending for local government as a whole net of income from certain fees and charges and certain specific grants. Net TSS is divided between the service blocks and sub-blocks (into "control totals"). The results of the SSA calculations for each service block are generally multiplied by a scaling factor which is calculated in such a way as to ensure that the SSA elements sum to the appropriate control total. The totals for the

sub-blocks are given in Appendix I, Table 1 (the column headed "net TSS"). The scaling factors are shown in Appendix I, Table 7.

The assessments for the service blocks are based on simple formulae using relevant indicators of the characteristics of each authority or area selected. These formulae and indicators were selected following extensive analysis, research and discussion with local authority representatives. Most formulae consist of a client group multiplied by the unit cost for the client group. Adjustments are made to some assessments to take account of the difference in the extra cost of providing a service which result from variations in additional needs (cost adjustment). Some assessments are further adjusted for the extra costs of employment and national non-domestic rates in certain parts of the country (area cost adjustment).

For a number of the assessments regression analysis has been used as a basis for the indicator weights. Regression is a statistical technique which can be used to investigate the relationship between a dependent variable (such as expenditure per head in each local authority) and a set of independent variables (such as density of population or the proportion of people in an authority receiving income support).

[1]*Full details of the NDR and RSG calculations at authority level are in The Local Government Finance Report (England) 1997/98. The RSG calculations are more complex in the London area. This is because the Metropolitan Police provides probation and magistrates' courts services in inner London but not in the remainder of the Metropolitan Police District.*

APPENDIX I TABLE 1 RELATIONSHIP BEWEEN GROSS TSS AND CONTROL TOTALS

Gross Total Standard Spending (TSS) for 1997/98 is £45.484 billion, including £325 million for the Community Care special grant, but excluding £150m of restructuring costs which are expected to be met by credit approvals and also excluding £6.5m in respect of the City Offset and £24.8 million in respect of Specified Bodies. The table shows how this is broken down by service block and sub-block. Net TSS corresponds to the SSA control totals.

1997/98 STANDARD SPENDING CONTROL TOTALS (£m)

Service	**1997/98 Gross TSS**	**1997/98 Specific Grants***	**1997/98 Net TSS**
Primary	7830.3	131.2	7699.1
Secondary	7778.3	85.4	7692.9
Post-16	1139.3	21.3	1118.0
Under 5	568.8	6.8	562.0
Other	772.9	4.7	768.2
Education	**18089.6**	**249.5**	**17840.2**
Children	1782.9	28.2	1754.7
Elderly residential	2809.6	251.3	2558.3
Elderly domiciliary	1668.5	68.6	1599.9
Other social services	1585.0	105.8	1479.2
Personal social services	**7846.0**	**453.9**	**7392.1**
Police	**6390.2**	**3301.6**	**3088.6**
Fire	**1237.2**	**0.0**	**1237.2**
Highway maintenance	**1759.0**	**0.0**	**1759.0**
District level	5311.8	53.1	5258.7
County level	2360.0	594.2	1765.8
Rent allowance payments	310.0	0.0	310.0
Housing benefit administration	159.7	79.7	80.0
Rail	173.5	173.5	0.0
Flood defence	215.5	0.0	215.5
Coast protection	9.9	0.0	9.9
National Parks and Broads	21.7	15.9	5.8
Other Services excluding interest receipts	**8562.1**	**916.4**	**7645.7**
Interest receipts	436.0	0.0	436.0
Other Services including interest receipts	**8126.1**	**916.4**	**7209.7**
Debt charges	2299.9	0.0	2299.9
Interest on set-aside capital receipts	-264.0	0.0	-264.0
PH Projects	0.2	0.0	0.2
Capital Financing	**2036.1**	**0.0**	**2036.1**
Revenue Expenditure	**45484.4**	**4921.3**	**40563.0**

Note: due to rounding, rows and columns do not always sum to their total figures.
** including community care special grant*

APPENDIX I TABLE 7 SCALING FACTORS The scaling factors given below are those used in the calculation of SSA elements for each authority. In general they are applied to each authority's SSA at the final stage of the calculation, according to the formulae given in the Local Government Finance Report. The scaling factors are needed to ensure that the national total for each SSA element equals the control total for that element. The groups of authorities for Interest Receipts arid Interest on Capital Receipts are those referred to in paragraphs 4.55 and 4.65 of the Local Government Finance Report.

1997/98 SCALING FACTORS

1 EDUCATION	
Primary Education	0.99999879822669
Secondary Education	1.00000003118648
Post-16 Education	1.00001172195554
Under 5 Education	0.99996547721178
Other Education	0.99948971291252
2 PERSONAL SOCIAL SERVICES	
Children's PSS	0.97887492619293
Elderly Residential PSS	0.99995107102374
Elderly Domiciliary PSS	1.35630477026772
Other PSS	1.27723508576216
3 POLICE	
Police	0.98470485851971
4 FIRE	
Fire	0.99966108654872
5 HIGHWAY MAINTENANCE	
Highway maintenance	1.00024388049147
6 ALL OTHER SERVICES	
District Level Other Services	0.90546736287440
County Level Other Services	1.03882957238915
Rent allowance payments	0.77896157340150
Housing benefit administration	1.00054290945091
Flood Defence	1.05966140015061
Coast Protection	1.03828002097535
National Parks	1.05261992562333
Interest Receipts for group (i)	-0.03041165590181
Interest Receipts for group (ii)	-0.01011048106349
Interest Receipts for group (iii)	-0.01332829062654
Interest Receipts for group (iv)	-0.00914335260932
7 CAPITAL FINANCING	
Debt charges	0.85363614606465
Interest on Set-Aside Capital Receipts for group (i)	-0.06182368953819
Interest on Set-Aside Capital Receipts for group (ii)	-0.00428528367311
Interest on Set-Aside Capital Receipts for group (iii)	-0.01777704499677
Interest on Set-Aside Capital Receipts for group (iv)	-0.01096363553650
PFI Projects	1.00025006251563

Appendix B

Population Data

City / Town	Census Population				
	1951	1961	1971	1981	1991
Chester	82492	102053	115705	116592	115680
Bath	79294	80901	84670	80771	80689
Bournemouth	144845	154296	153869	145704	154677
Cheltenham	68687	72154	84433	99203	103566
Crewe and Nantwich	80056	91295	97197	98555	105400
Eastbourne	57821	60918	70921	77963	78778
Exeter	75509	80321	95729	96978	101395
Isle of Wight	95625	95752	109512	118594	127232
Leicester	285181	273470	284208	280284	272133
Norwich	121236	120096	122085	122890	122661
Oxford	98684	106291	108805	113653	124058
Warrington	115163	142113	162507	169372	185200
Winchester	70298	75027	85863	90826	96585
York	105371	104392	104782	99910	101436
	Census Population				
County	1951	1961	1971	1981	1991
Cheshire	824750	922146	866566	929961	940887
Cornwall	346442	342301	379242	432240	470912
Derbyshire	858832	877620	886611	910141	918648
Devon	797738	823751	898404	958745	1015969
Norfolk	548062	561071	625793	695683	742080

This data was collected from Census 1951, 1961, 1971, 1981, 1991, England and Wales, published by HMSO and held in the archive section of the public library in Bridgwater, Somerset.

Boundary changes and census modifications made the data collection difficult and,

although every effort was taken to make the data consistent, the accuracy for the figures for any area cannot be guaranteed.

Note also that the data for the graph of the government mid year estimates for Chester, figure 1.1, includes the students resident in Chester and hence differs from the data collected from the census figures shown in the table above.

Appendix C

Neural Network Programs

C.1 Automated Regularization

```
net=newff([0 6],[40 1],'tansig','purelin','trainbr');
    net.trainParam.epochs=300;
    p=[0 1 2 3 4];
    t=[1.44845 1.54296 1.53869 1.45704 1.54677];
    net=init(net);
    net=train(net,p,t);
    x=[0:.1:4];
    fa=sim(net,x);
    plot(0,1.44845,'*',1,1.54296,'*',2,1.53869,'*',3,1.45704,'*',4,1.54677,'*',x,fa)
```

C.2 Early Stopping

```
net=newff([0 6],[16 1],'tansig','purelin','traingdx');
    net.trainParam.epochs=300;
    p=[0 1 2 3 4];
    t=[1.44845 1.54296 1.53869 1.45704 1.54677];
    net=init(net);
    net=train(net,p,t);
    x=[0:.1:4];
    fa=sim(net,x);
    plot(0,1.44845,'*',1,1.54296,'*',2,1.53869,'*',3,1.45704,'*',4,1.54677,'*',x,fa)
```

C.3 Generalized Regression

```
p=[0 1 2 3 4];
    t=[1.44845 1.54296 1.53869 1.45704 1.54677];
    net=newgrnn(p,t);
    x=[0:.1:5];
    fa=sim(net,x);
    plot(0,1.44845,'*',1,1.54296,'*',2,1.53869,'*',3,1.45704,'*',4,1.54677,'*',x,fa)
```

C.4 Gradient Descent with Momentum

C.4.1 4 by R^2 Input

```
net=newff([0 3;0 3;0 3;0 3],[3,1],'tansig','purelin','traingdm');
    net.trainParam.show = 50;
    net.trainParam.lr = 0.05;
    net.trainParam.mc = 0.9;
    net.trainParam.epochs = 1000;
    net.trainParam.goal = 1e-5;
    p = [.70921 .95729 1.09512 2.84208;.77963 .96978 1.18594 2.80284];
    t = [.78778 1.01395 1.27232 2.72133];
    pt = [1.15705 .84670 1.53869 .84433;1.16592 .80771 1.45704 .99203 ];
    net = train(net,p,t);
    t = [.78778 1.01395 1.27232 2.72133]
    ap = sim(net,p)
    tt = [1.15680 .80689 1.54677 1.03566]
    apt = sim(net,pt)
```

C.4.2 4 by R^4 Input

```
net=newff([0 3;0 3;0 3;0 3],[7,1],'tansig','purelin','traingdm');
    net.trainParam.show = 50;
    net.trainParam.lr = 0.05;
    net.trainParam.mc = 0.9;
    net.trainParam.epochs = 100000;
    net.trainParam.goal = 1e-8;
    p = [.57821 .75509 .95625 2.85181;.60918 .80321 .95792 2.73470;
    .70921 .95729 1.09512 2.84208;.77963 .96978 1.18594 2.80284];
    t = [.78778 1.01395 1.27232 2.72133];
    pt = [.82492 .79294 1.44845 .68687;1.02053 .84670 1.54296 .72154;
```

```
1.15705 .84670 1.53869 .84433;1.16592 .807711.45704 .99203 ];
net = train(net,p,t);
t = [.78778 1.01395 1.27232 2.72133]
ap = sim(net,p)
tt = [1.15680 .80689 1.54677 1.03566]
apt = sim(net,pt)
```

C.4.3 6 by R^4 Input

```
net=newff([0 3;0 3;0 3;0 3],[12,1],'tansig','purelin','traingdm');
    net.trainParam.show = 50;
    net.trainParam.lr = 0.05;
    net.trainParam.mc = 0.9;
    net.trainParam.epochs = 100000;
    net.trainParam.goal = 1e-8;
    p = [.57821 .75509 .95625 2.85181 1.21236 .98684;
    .60918 .80321 .95792 2.73470 1.20098 1.06291;
    .70921 .95729 1.09512 2.84208 1.22085 1.08805;
    .77963 .96978 1.18594 2.802841.22890 1.13653];
    t = [.78778 1.01395 1.27232 2.72133 1.22661 1.24058];
    pt = [.82492 .79294 1.44845 .68687 .80056 1.05371;
    1.02053 .84670 1.54296 .72154 .91295 1.04392;
    1.15705 .84670 1.53869 .84433 .97197 1.04782;
    1.16592 .80771 1.45704 .99203 .98555 .99910];
    net = train(net,p,t);
    t = [.78778 1.01395 1.27232 2.72133 1.22661 1.24058]
    ap = sim(net,p)
    tt = [1.15680 .80689 1.54677 1.03566 1.054001.01436]
    apt = sim(net,pt)
```

C.5 Levenberg-Marquardt

```
net=newff([0 3;0 3;0 3;0 3],[15,1],'tansig','purelin','trainlm');
    net.trainParam.show = 50;
    net.trainParam.epochs = 10000;
    net.trainParam.goal = 1e-10;
    p = [.57821 .75509 .95625 2.85181 1.21236 .98684;
    .60918 .80321 .95792 2.73470 1.20098 1.06291;
    .70921 .95729 1.09512 2.84208 1.22085 1.08805;
    .77963 .96978 1.18594 2.802841.22890 1.13653];
```

```
t = [.78778 1.01395 1.27232 2.72133 1.22661 1.24058];
pt = [.82492 .79294 1.44845 .68687 .80056 1.05371;
1.02053 .84670 1.54296 .72154 .91295 1.04392;
1.15705 .84670 1.53869 .84433 .97197 1.04782;
1.16592 .80771 1.45704 .99203 .98555 .99910];
net = train(net,p,t);
t = [.78778 1.01395 1.27232 2.72133 1.22661 1.24058]
ap = sim(net,p)
tt = [1.15680 .80689 1.54677 1.03566 1.054001.01436]
apt = sim(net,pt)
```

Bibliography

[1] Burden, R. L. and Faires, J. D. *Numerical Analysis*, Boston, PWS-Kent Publishing Company, 1989.

[2] Demuth, H. and Beale, M. *Neural Network Toolbox for use with MATLAB, User Guide, Version 3*, Natick, The Math Works Inc., 1998.

[3] Evans, G. *Practical Numerical Analysis*, Chichester, Wiley, 1995.

[4] Harding, R. D. and Quinney, D. A. *A Simple Introduction to Numerical Analysis, Volume 2: Interpolation and Approximation*, Bristol, Adam Hilger, 1989.

[5] Ralston, A. and Rabinowitz, P. *A First Course in Numerical Analysis*, New York, McGraw-Hill Book Company, 1978.

[6] Occasional Paper 37, Making a Population Estimate, Office of Population Censuses and Surveys,1991.

[7] Standard Spending Assessment 1997/98, LGD Division, Department of the Environment.

[8] Unit 3, MST204 Mathematical Models and Methods, The Open University,1991.